(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-531878
(P2004-531878A)
(43)公表日　平成16年10月14日(2004.10.14)

(51)Int.Cl.[7]	F I	テーマコード（参考）
H01L 21/02	H01L 21/02 Z	

審査請求 未請求　予備審査請求 有　（全 42 頁）

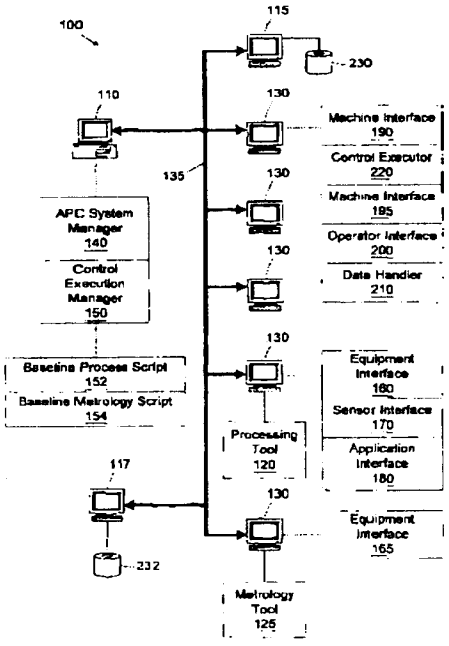
(21)出願番号	特願2002-568120(P2002-568120)
(86)(22)出願日	平成13年10月22日(2001.10.22)
(85)翻訳文提出日	平成15年8月21日(2003.8.21)
(86)国際出願番号	PCT/US2001/050178
(87)国際公開番号	WO2002/069063
(87)国際公開日	平成14年9月6日(2002.9.6)
(31)優先権主張番号	09/789,871
(32)優先日	平成13年2月21日(2001.2.21)
(33)優先権主張国	米国(US)

(71)出願人　591016172
アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド
ADVANCED MICRO DEVICES INCORPORATED
アメリカ合衆国、９４０８８－３４５３カリフォルニア州、サニィベイル、ピィ・オゥ・ボックス・３４５３、ワン・エイ・エム・ディ・プレイス、メイル・ストップ・６８（番地なし）
(74)代理人　100099324
弁理士　鈴木　正剛
(74)代理人　100111615
弁理士　佐野　良太

最終頁に続く

(54)【発明の名称】ベースライン制御スクリプトを用いてツールを制御するための方法および装置

(57)【要約】
製造システム（１００）を制御するための方法が、複数のツール（１２０、１２５）で被加工物を処理し、この複数のツール（１２０、１２５）のうちの選択したツール（１２０、１２５）についてベースライン制御スクリプト（１５２、１５４）を開始し、ベースライン制御スクリプト（１５２、１５４）のコンテキスト情報を提供し、コンテキスト情報に基づいてツールのタイプを判定し、選択したツール（１２０、１２５）の制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツール（１２０、１２５）のコントロールアクションを生成することを含む。製造システム（１００）は、被加工物を処理するように作られた複数のツール（１２０、１２５）と、制御実行マネージャ（１５０）と、コントロールエグゼキュータ（２２０）とを含む。制御実行マネージャ（１５０）は、複数のツール（１２０、１２５）のうちの選択したツール（１２０、１２５）についてベースライン制御スクリプト（１５２、１５４）を開始し、ベースライン制御スクリプト（１５２、１５４）のコンテキスト情報を提供するように作ら



【特許請求の範囲】
【請求項１】
複数のツール（１２０、１２５）で被加工物を処理する際に、
前記複数のツール（１２０、１２５）のうちの選択したツール（１２０、１２５）についてベースライン制御スクリプト（１５２、１５４）を開始する処理と、
前記ベースライン制御スクリプト（１５２、１５４）のコンテキスト情報を提供する処理と、
前記コンテキスト情報に基づいてツールのタイプを判定する処理と、
前記選択したツール（１２０、１２５）の制御ルーチンをツールのタイプに基づいて選択する処理と、
前記制御ルーチンを実行して前記選択したツール（１２０、１２５）のコントロールアクションを生成する処理とを含む、製造システム（１００）を制御するための方法。
【請求項２】
前記制御ルーチンを選択する処理が、制御ルーチンのライブラリ（２４０）にリンクする処理をさらに含む、請求項１に記載の方法。
【請求項３】
前記コンテキスト情報が前記選択したツールに関連したエンティティ識別コードを含み、前記ツールのタイプを判定する処理が前記エンティティ識別コードに基づいてツールのタイプを判定する処理をさらに含む、請求項１に記載の方法。
【請求項４】
前記コンテキスト情報がオペレーション識別コードを含み、前記制御ルーチンを選択する処理が前記ツールのタイプと前記オペレーション識別コードとに基づいて制御ルーチンを選択する処理をさらに含む、請求項１または３に記載の方法。
【請求項５】
前記コンテキスト情報が製品識別コードを含み、前記制御ルーチンを選択する処理がツールのタイプと前記製品識別コードとに基づいて制御ルーチンを選択する処理をさらに含む、請求項１、３または４に記載の方法。
【請求項６】
被加工物を処理するように作られた複数のツール（１２０、１２５）と、
前記複数のツール（１２０、１２５）のうちの選択したツール（１２０、１２５）についてベースライン制御スクリプト（１５２、１５４）を開始し、前記ベースライン制御スクリプト（１５２、１５４）のコンテキスト情報を提供するように作られた制御実行マネージャ（１５０）と、
前記ベースライン制御スクリプト（１５２、１５４）を実行し、前記コンテキスト情報に基づいてツールのタイプを判定し、前記選択したツール（１２０、１２５）の制御ルーチンをツールのタイプに基づいて選択し、前記制御ルーチンを実行して前記選択したツール（１２０、１２５）のコントロールアクションを生成するように作られたコントロールエグゼキュータ（２２０）とを含む、製造システム（１００）。
【請求項７】
前記コントロールエグゼキュータ（２２０）が制御ルーチンのライブラリ（２４０）にリンクして制御ルーチンを選択するように作られている、請求項６に記載のシステム（１００）。
【請求項８】
前記コンテキスト情報が前記選択したツールに関連したエンティティ識別コードを含み、コントロールエグゼキュータ（２２０）が前記エンティティ識別コードに基づいてツールのタイプを判定するように作られている、請求項６に記載のシステム（１００）。
【請求項９】
前記コンテキスト情報がオペレーション識別コードを含み、前記コントロールエグゼキュータ（２２０）がツールのタイプと前記オペレーション識別コードとに基づいて制御ルーチンを選択するように作られている、請求項６または８に記載のシステム（１００）。

【請求項１０】
前記コンテキスト情報が製品識別コードを含み、前記コントロールエグゼキュータ（２２０）がツールのタイプと前記製品識別コードとに基づいて制御ルーチンを選択するように作られている、請求項６、８または９に記載のシステム（１００）。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は広義には半導体デバイス製造の分野に関し、特に、ベースライン制御スクリプトを用いてツールを制御するための方法および装置に関する。
【背景技術】
【０００２】
半導体業界では、マイクロプロセッサ、メモリデバイスなどの集積回路デバイスの品質、信頼性、スループットを向上させたいという強い要求が常にある。この要求は、それまでよりも確実に動作する高品質のコンピュータや電子デバイスを求める消費者需要を受けてなお一層高まっている。こうした需要があることから、トランジスタなどの半導体デバイスの製造ならびにこのようなトランジスタを組み入れた集積回路デバイスの製造の点でたゆみない改善がなされてきている。また、典型的なトランジスタの構成部品の製造時に欠陥が少なくなることで、トランジスタ１つあたりの総コストの削減になるほか、このようなトランジスタを組み入れた集積回路デバイスのコストの削減にもなる。
【０００３】
通常、フォトリソグラフィ用ステッパ、エッチツール、成膜ツール、研磨ツール、高速熱処理ツール、インプランテーションツールなどを含むさまざまな処理ツールを利用して、一連の処理ステップが多数のウエハ上で実施される。ここ数年、半導体処理ツールの基礎をなすいくつかの技術が次第に注目されるようになり、相当な改良がなされている。しかしながら、この分野でこうした進歩があったにもかかわらず、現段階で商用入手可能な処理ツールの多くにはある種の欠点がある。特に、このようなツールには、ユーザが利用しやすい形式でパラメータデータの履歴を提供する機能や、イベントログの記録、最新の処理パラメータとラン（実行）全体の処理パラメータのリアルタイムでのグラフィック表示、遠隔地すなわちローカルサイトおよび世界中で監視を行う機能などの高度なプロセスデータ監視能力が欠けていることが多い。これらの欠点が原因で、スループットや精度、安定性、再現性などの重要な処理パラメータ、処理温度、機械的なツールのパラメータなどが最適とはいえない形で制御される結果になりかねない。この多様性はラン内での格差、ランとランとの格差、ツールとツールとの格差として表れ、こうした格差が進むと製品の品質や性能の偏りへとつながる可能性があるのに対し、これらのツールのための理想的な監視・診断システムがあれば重要なパラメータの制御を最適化するための手段が得られるだけでなく上記の多様性を監視する手段も得られるであろう。
【０００４】
半導体処理ラインのオペレーションを改善するためのひとつの手法として全工場規模の制御システムを採用してさまざまな処理ツールのオペレーションを自動制御することがあげられる。製造ツールが製造フレームワークと通信または処理モジュールのネットワークと通信する。それぞれの製造ツールは一般に機器インタフェースと接続されている。機器インタフェースは、製造ツールと製造フレームワークとの間の通信をしやすくするマシンインタフェースと接続されている。マシンインタフェースは通常、高度プロセス制御（ＡＰＣ）システムの一部であってもよいものである。ＡＰＣシステムは製造モデルに基づいて制御スクリプトを開始するが、この制御スクリプトは製造プロセスを実行するのに必要なデータを自動的に取り込むソフトウェアプログラムであってもよい。しばしば半導体デバイスは複数の製造ツールによって複数のプロセスで段階をおって作られ、処理後の半導体デバイスの品質に関係するデータが生成される。
【０００５】
製作プロセスの間、製作対象となるデバイスの性能に影響するさまざまな出来事が起こる

可能性がある。つまり、製作プロセスステップのばらつきが原因でデバイス性能のばらつきが発生する。構造の臨界寸法、ドーピングレベル、接触抵抗、粒子汚染などの要因がすべてデバイスの最終性能に影響する可能性を持つことがある。処理ラインのツールはそれぞれ処理のばらつきを減らすべく性能モデルに従って制御されている。普通に制御されるツールとしては、フォトリソグラフィ用ステッパ、研磨ツール、エッチングツール、成膜ツールがあげられる。これらのツール用のプロセスコントローラには前処理および／または後処理測定データが供給される。処理時間などの動作レシピパラメータについては、性能モデルと計測情報をもとにプロセスコントローラで計算し、標的値にできるだけ近い処理結果を達成しようと試みられている。このようにしてばらつきを減らすことで、スループットを向上させ、コストを削減し、デバイス性能をそれまでよりも高くできるなどの結果につながるが、これらはいずれも収益性の向上に結び付く。

【発明の開示】

【発明が解決しようとする課題】

【０００６】

コンフィギュレーション制御と効率の問題は、全工場規模のＡＰＣシステムなどの分散コンピューティング環境では普通に見られるものである。一般に、プロセスコントローラを構築するための制御コードは多くのソフトウェア開発者によって書かれている。個々の開発者が特定タイプのコントローラを開発しつつ広範囲にわたって作業をすることもある。彼ら開発者がそれぞれ独自のプログラミングスタイルを持ち、自分で作り出したルーチンに頼っていることは珍しくない。たとえば、ＡＰＣフレームワーク内のデータベースや他のエンティティと接続し、さまざまな数学関数や基本のユーティリティ関数を実行するためのルーチンのセットを開発者ごとに持っている場合がある。

【０００７】

このような形態をとることに伴うひとつの問題として、プロセス制御スクリプト間の一貫性がほとんどない点があげられる。開発者独自のスクリプトが多数あることもコンフィギュレーション制御の問題や効率面での問題の原因となる。開発者らは、すでに別の開発者が違うタイプのプロセスコントローラ向けに開発したであろうものと同様のコードを書くのに相当な時間を費やす場合がある。標準から外れたコードのデバッグにもさらに多くの時間を要し、余計に効率が悪くなる。

【０００８】

本発明は、上述した問題のうちひとつまたはそれ以上の影響を解決するか少なくとも低減しようとするものである。

【課題を解決するための手段】

【０００９】

発明の開示

本発明の一態様は、製造システムを制御するための方法に見られる。この方法は、複数のツールで被加工物を処理し、この複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始し、ベースライン制御スクリプトのコンテキスト情報を提供し、このコンテキスト情報に基づいてツールのタイプを判定し、選択したツールの制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツールのコントロールアクションを生成することを含む。

【００１０】

本発明のもうひとつの態様は、被加工物を処理するように作られた複数のツールと、制御実行マネージャと、コントロールエグゼキュータとを含む製造システムに見られる。制御実行マネージャは、複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始し、ベースライン制御スクリプトのコンテキスト情報を提供するように作られている。コントロールエグゼキュータは、ベースライン制御スクリプトを実行し、このコンテキスト情報に基づいてツールのタイプを判定し、選択したツールの制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツールのコントロールアクションを生成するように作られている。

【００１１】
添付の図面を参照して以下の説明を参照することで本発明を理解することができよう。図中、同様の構成要素には同様の参照符号を付してある。
【００１２】
本発明はさまざまな形に改変および変更が可能なものであるが、一例としてその特定の実施形態を図示し、本願明細書中にて詳細に説明してある。しかしながら、特定の実施形態に関する本願明細書の記載がここに開示の特定の形態に本発明を限定することを意図したものではなく、添付の特許請求の範囲に規定の本発明の範囲に含まれる改変例、等価物および代わりとなる物をすべて包含することを意図している点を理解されたい。
【発明を実施するための最良の形態】
【００１３】
発明の実施の形態
以下、本発明の実施形態について説明する。明瞭な説明とするために、本願明細書では実際に使われる実装の構造すべてについて説明してあるとは限らない。もちろん、このような実際の実施形態を開発するにあたって、実装ごとに異なるシステム関連の制約およびビジネス関連の制約に従うなど、実装ごとに固有の多数の決定を行い、開発者ごとに異なる目標を達成しなければならない点は理解できよう。さらに、このような開発の取り組みが複雑で時間を要するものとなる可能性があるとはいえ、本願開示の利益を享受する当業者の日常作業になろうものであることも理解できよう。
【００１４】
以下、図面を参照するが、まず図１を参照すると、高度プロセス制御（ＡＰＣ）システム１００の簡略ブロック図が示されている。ＡＰＣシステム１００は、ランとランとの間の制御や故障の検出／分類を可能にしている取り替え可能な標準化ソフトウェアコンポーネントからなる分散ソフトウェアシステムである。これらのソフトウェアコンポーネントは、半導体製造装置・材料に関する業界団体（ＳＥＭＩ）コンピュータ統合生産（ＣＩＭ）フレームワーク対応のシステム技術仕様および高度プロセス制御（ＡＰＣ）フレームワークに基づくアーキテクチャ標準を満たしている。ＣＩＭ（ＣＩＭフレームワークドメインアーキテクチャに関するＳＥＭＩ　Ｅ８１－０６９９－暫定仕様書）およびＡＰＣ（ＣＩＭフレームワーク高度プロセス制御コンポーネントに関するＳＥＭＩ　Ｅ９３－０９９９－暫定仕様書）の各仕様についてはＳＥＭＩから一般に入手可能である。この特有のアーキテクチャは、オブジェクト指向プログラミングを利用しているソフトウェアに極めて大きく依存し、分散オブジェクトシステム向けにオブジェクトマネジメントグループ（ＯＭＧ）が策定した共通オブジェクトリクエストブローカーアーキテクチャ（ＣＯＲＢＡ）およびＣＯＲＢＡサービスの仕様を採用している。このＯＭＧ　ＣＯＲＢＡアーキテクチャについての情報および仕様は容易に一般に入手可能である。本願明細書にて説明するようなＡＰＣシステム１００の機能を果たすようにすることのできるソフトウェアシステムの一例として、ケーエルエー・テンコール・インコーポレイテッド（ＫＬＡ－Ｔｅｎｃｏｒ，Ｉｎｃ．）から提供されているＣａｔａｌｙｓｔシステムがあげられる。
【００１５】
これらのコンポーネントはＣＯＲＢＡインタフェース定義言語（ＩＤＬ）を使って互いに通信し、その対話をサポートするための共通のサービスセットに頼っている。分散オブジェクトサービスの標準セットはＯＭＧによって規定されている。これらのサービスには以下のものがある。
【００１６】
ＣＯＲＢＡ－コンポーネントとコンポーネントとの間での直接対話に必ず使われる標準ベースの通信プロトコルである。オブジェクト指向の遠隔呼出し通信モデルに従って標準インタフェースを定義することができる。これらのインタフェースとすべてのＡＰＣ通信はＩＤＬを使って定義される。コンポーネントは互いに他のインタフェースでオペレーションを呼び出すことによって通信する。コンポーネント間ではオペレーションパラメータおよび返り値としてデータの受け渡しがなされる。

【００１７】
ＯＭＧイベントサービスーコンポーネント間での非同期的な通信をサポートするものである。ＡＰＣオブジェクトの多くは状態が変化したときにイベントを発信する。これらのイベントは、該当するイベントサブスクライバによって受信される。ＡＰＣシステム内でのイベント利用の例としては、通信コンポーネントの状態（エラー状態を含む）、故障の検出および分類ソフトウェアで検出した故障アラームの通知、マシンのステータスと収集したデータの報告があげられるが、これに限定されるものではない。
【００１８】
ＯＭＧトレーディングサービスーコンポーネントが自己と対話すべき別のコンポーネントを見つけることができるようにするものである。コンポーネントをインストールすると、そのサービスの説明（サービスのオファー）がトレーディングサービスにエクスポートされる。別のコンポーネントが特定の規準を満たすサービスプロバイダの一覧を後から要求することができる。トレーディングサービスでは、要求されたサービスを提供できる他のコンポーネントの一覧を提供する。この機能は、コンポーネントの起動時にひとつのコンポーネントが自己と対話しなければならない他のコンポーネントを見つけることができるようにする目的で使われる。また、これはプラン実行コンポーネントがプランに指定された必要な能力を提供するのにケイパビリティプロバイダ（Ｃａｐａｂｉｌｉｔｙ Ｐｒｏｖｉｄｅｒ）を見つけなければならないプランスタートアップ（Ｐｌａｎ Ｓｔａｒｔｕｐ）時にも使われる。
【００１９】
これらのサービスは従来技術において周知である。ＯＭＧのＣＯＲＢＡ／ＩＩＯＰ仕様の文書およびＣＯＲＢＡサービス仕様の文書は当業者に広く配布されており、そこにはさらに詳細な説明がなされている。
【００２０】
図示の実施形態では、ＡＰＣシステム１００は半導体製造環境を制御できるように作られている。複数のコンポーネントがＣＯＲＢＡインタフェース定義言語（ＩＤＬ）を使って互いに通信する。連携しているソフトウェアコンポーネントは、プロセス制御プラン／ストラテジーを管理し、プロセス機器、計測ツール、アドオンセンサからデータを収集し、この情報を利用してさまざまなプロセス制御アプリケーション／アルゴリズムを呼び出し、プロセスモデルを更新し、該当する場合はツール動作レシピパラメータを修正／ダウンロードする。ＡＰＣシステム１００は半導体生産プロセスを制御するための全工場規模のソフトウェアシステムであるが、これは本発明を実施するにあたって必要なことではない。本発明が教示するストラテジーは、任意の規模で異なるコンピュータシステムに適用可能なものである。
【００２１】
代表的な実装例では、ＡＰＣシステム１００は、ＡＰＣホストコンピュータ１１０と、データベースサーバ１１５、１１７と、処理ツール１２０と、計測ツール１２５と、１つまたはそれ以上のワークステーション１３０とを含む。ＡＰＣシステムの各コンポーネントはバス１３５を介して相互に接続されている。バス１３５は実際は複数の層で構成されて複数のプロトコルを利用するものであってもよい。ＡＰＣシステム１００の全体としてのオペレーションはＡＰＣホストコンピュータ１１０上に常駐するＡＰＣシステムマネージャ１４０によって指示されている。ＡＰＣシステムマネージャ１４０は、ＡＰＣフレームワーク用に開発されたすべてのサーバの管理サービス、コンフィギュレーションサービス、イベントサービス、状態サービス；ＡＰＣシステム１００内のコンポーネントの定義、グルーピング、インストール、マネジメント；診断および監視の目的でアクティビティおよびトレース情報を捕捉するための中央サービス；セットアップ値、システム環境設定を含むコンポーネントコンフィギュレーション情報の中央の収納場所；従属オブジェクトおよびイベントチャネルのリストを提供するものである。しかしながら、別の実施形態では、これらの機能を、ベースマネージャ、システムマネージャ、ロガー、レジストリなどの１つまたはそれ以上のソフトウェアコンポーネントに分割してもよい。

【００２２】
ＡＰＣシステム１００は複数の処理モジュールからなるネットワークを含む。これらの処理モジュールは「インテグレーションコンポーネント」と呼ばれることもある。インテグレーションコンポーネントは、既存のファクトリーシステムとのインタフェースとして機能し、ＡＰＣプランを走らせることができるようにするものである。「ＡＰＣプラン」とは、詳細については後述するように特定のタスクを実行するために呼ばれるアプリケーションプログラムのことである。インテグレーションコンポーネントについては、ＡＰＣシステム１００内のさまざまな処理資源によって受け入れられる場合があるものとして示してある。これらの特定のホスティングロケーションは例示の目的で提供されている。処理リソース同士は相互に接続されており、システムの複雑さに応じてさまざまなソフトウェアコンポーネントをさまざまなコンピュータに分散させてもよいし、中央に集中させてもよい。この特定の実施形態におけるインテグレーションコンポーネントは各々ソフトウェア的に実装されている。これらのコンポーネントは従来技術において周知のオブジェクト指向プログラミングの手法を使ってＣ＋＋でプログラムされている。ＡＰＣシステム１００の利点のひとつにモジュール構造であるという点があるが、これによってソフトウェアコンポーネントの移植性が得られる。インテグレーションコンポーネントは、ＡＰＣシステムマネージャ１４０、制御実行マネージャ１５０、ツール１２０、１２５に関連した機器インタフェース１６０、１６５、処理ツール１２０に関連したセンサインタフェース１７０、アプリケーションインタフェース１８０、マシンインタフェース１９０、１９５、オペレータインタフェース２００、データハンドラ２１０を含むがこれに限定されるものではない。

【００２３】
制御実行マネージャ１５０は、ＡＰＣシステム１００のオペレーションの「振り付け」を主に担うコンポーネントである。制御実行マネージャ１５０はＡＰＣプランを解釈し、メインスクリプトおよびサブスクリプトを実行し、イベントの命令時にイベントスクリプトを呼び出す。多様なプラン、スクリプト、サブスクリプトをさまざまな実装に用いることができる。さまざまなプラン、スクリプト、サブスクリプトの具体的な数と機能はそれぞれの実装内容に依存する。たとえば、本実施形態は、
データ収集プラン－特定の処理機器からどのデータを収集すべきか、そのデータをどのようにして折り返し報告するかという要件を定義するセンサおよびマシンインタフェースで用いられるデータ構造
期間プラン－データ収集の開始、データ収集の終了など、センサを作動させるトリガー条件とトリガー遅延とを定義するプラン
レポーティングプラン－収集したデータをどのようにすべきかとデータの可用性をいつ信号伝達すべきかを定義するプラン
サンプリングプラン－外部センサによってデータを収集する頻度を定義するプラン
制御プラン－ＡＰＣアクティビティを実行するために一緒に使われるよう設計された制御スクリプトの集合
制御スクリプト－定義された特定の状況下でＡＰＣシステムが実行する一連のアクション／アクティビティ
などのプランを含むがこれに限定されるものではない。

【００２４】
制御実行マネージャ１５０は、処理ツール１２０などの特定のツールについて、すべてのインテグレーションコンポーネント内でユーザが定義したプロセス制御プランの実行をコーディネートする。指示があると、制御実行マネージャ１５０はプランとこれに関連したスクリプトとを読み出す。次に、サブスクリプトを前処理してメインスクリプトおよびイベントスクリプトへのルーチンを提供する。また、プランに指定されている、プランを実行するのに必要な能力の一覧を取得し、必要な能力が得られるしかるべきインテグレーションコンポーネントに接続する。

【００２５】

次に、制御実行マネージャ１５０は、このプランを走らせる責任をコントロールエグゼキュータ２２０に委ねる。図示の実施形態では、制御実行マネージャ１５０はベースライン制御スクリプトを使って実施すべきコントロールアクションを判定する。ベースラインプロセススクリプト１５２は処理ツール１２０などの処理ツールと併用すべきものであり、ベースライン計測スクリプト１５４は計測ツール１２５などの計測ツールと併用すべきものである。ベースラインスクリプト１５２、１５４の詳細については、図２乃至図６を参照して後述する。

【００２６】

制御実行マネージャ１５０は、プランを順次実行してプランの終了またはプラン実行時におけるエラーを制御実行マネージャ１５０に折り返し報告するためのコントロールエグゼキュータ２２０を、適当なベースラインプロセススクリプト１５２またはベースライン計測スクリプト１５４に基づいて作成する。このように、制御実行マネージャ１５０が実行されるすべてのプランの全体としてのマネジメントを担っているのに対し、各コントロールエグゼキュータ２２０はひとつのプランを走らせることだけを担っている。コントロールエグゼキュータ２２０は制御実行マネージャ１５０によって作成され、プランが生きている間は存在し、そのプランが終了または異常終了した旨の報告の後で制御実行マネージャ１５０によって破棄される。制御実行マネージャ１５０は、複数のコントロールエグゼキュータ２２０を介して複数のプランを同時に始めることができる。

【００２７】

マシンインタフェース１９０、１９５は、ＡＰＣシステムマネージャ１４０などのＡＰＣフレームワークと機器インタフェース１６０、１６５との間のギャップを埋めるものである。マシンインタフェース１９０、１９５は、処理ツールまたは計測ツール１２０、１２５とＡＰＣフレームワークとを接続し、マシンのセットアップ、起動、監視、データ収集をサポートする。この特定の実施形態では、マシンインタフェース１９０、１９５は主に機器インタフェース１６０、１６５の特定の通信とＡＰＣフレームワークのＣＯＲＢＡ通信との間の翻訳を行う。特に、マシンインタフェース１９０、１９５は、コマンド、ステータスイベント、収集されたデータを機器インタフェース１６０、１６５から受け取り、必要に応じてこれを他のＡＰＣコンポーネントおよびイベントチャネルに転送する。今度は、他のＡＰＣコンポーネントからの応答がマシンインタフェース１９０、１９５で受信され、機器インタフェース１６０、１６５に送られる。また、マシンインタフェース１９０、１９５は、必要に応じてメッセージとデータを再フォーマットおよび再構築する。マシンインタフェース１９０、１９５は、ＡＰＣシステムマネージャ１４０内でのスタートアップ／シャットダウン手順をサポートする。また、機器インタフェース１６０、１６５によって収集されたデータを一時的に蓄積し、適切なデータ収集イベントを発信するＡＰＣデータコレクタとしても機能する。

【００２８】

センサインタフェース１７０は、処理ツール１２０のオペレーションを監視しているセンサが生成したデータを収集する。センサインタフェース１７０は、ラボビュー（ＬａｂＶＩＥＷ）または他のセンサなどの外部センサやバスをベースにしたデータ取得ソフトウェアと通信するための適当なインタフェース環境を提供するものである。アプリケーションインタフェース１８０は、ラボビュー（ＬａｂＶＩＥＷ）、マセマティカ（Ｍａｔｈｅｍａｔｉｃａ）、モデルウェア（ＭｏｄｅｌＷａｒｅ）、マトラボ（ＭａｔＬａｂ）、シムカ（Ｓｉｍｃａ）４０００、エクセル（Ｅｘｃｅｌ）などの制御プラグインアプリケーションを実行するための適切なインタフェース環境を提供するものである。センサについては相手先商標製品製造（ＯＥＭ）によって処理ツール１２０と一緒に供給してもよいし、ＯＥＭから入手した後でインストールされる「アドオン」センサであってもよい。センサインタフェース１７０はセンサによって生成されたデータを収集する。このセンサは、たとえば動作条件の圧力および温度についてのデータを生成するものであっても構わない。アプリケーションインタフェース１８０はコントロールエグゼキュータ２２０からデータを得て、そのデータに関する計算または分析を実行する。次に、結果をコントロールエグ

ゼキュータ２２０に返す。マシンインタフェース１９０およびセンサインタフェース１７０は、使われるデータを収集するための共通の機能一式を利用する。機器インタフェース１６０は処理ツール１２０についてセンサが収集したそれぞれのデータを集め、集めたデータをマシンインタフェース１９０に送信する。

【００２９】

オペレータインタフェース２００は、グラフィカルユーザインタフェース（ＧＵＩ）（図示せず）によってウエハ製作技術者とＡＰＣシステム１００との通信を容易にするものである。このＧＵＩはＷｉｎｄｏｗｓ（Ｒ）またはＵＮＩＸベースのオペレーティングシステムであっても構わないが、これは本発明を実施するにあたって必要なことではない。特に、いくつかの別の実施形態ではＧＵＩを採用することすらせず、ディスクオペレーティングシステム（ＤＯＳ）ベースのオペレーティングシステムを使って通信を行うことができる。オペレータインタフェース２００はダイアログボックスを表示して情報を提供し、指示を求め、別のデータを収集する。ＣＯＲＢＡインタフェースによって、技術者らはオペレータインタフェース２００のコンポーネントを使って任意の数のディスプレイ群に多様なポップアップダイアログを同時に表示することができる。また、オペレータインタフェース２００は、ポップアップを表示させることのできるディスプレイ群も維持する。さらに、オペレータインタフェース２００は、アナウンスメントオペレーションすなわちメッセージと「ＯＫ」ボタンだけの単純なポップアップを表示する一方的なメッセージを提供する。

【００３０】

データハンドラ２１０は、他のＡＰＣシステム１００のコンポーネントが集めたデータを受け取り、これらのデータをデータベースサーバ１１５、１１７上のデータストア２３０、２３２（リレーショナルデータベースなど）に格納する。データハンドラ２１０は、標準的な構造化照会言語（ＳＱＬ）コマンドを受け取ることのできるものであってもよいし、あるいは、データハンドラ２１０は、異なるタイプのアクセスプロトコルを翻訳してＳＱＬコマンドまたは他の何らかのプロトコルコマンドを生成するものであってもよい。データ格納機能を中央に集めることで、さまざまなコンポーネントの移植性が高くなる。

【００３１】

次に、図２に示す簡略ブロック図を参照してベースライン制御スクリプト１５２、１５４の大まかなオペレーションについて説明する。同図において、ベースライン制御スクリプト１５２、１５４と多様な共有ベースラインライブラリとの間のリンクが示されている。総じて言うと、ベースライン制御スクリプト１５２、１５４は、ＡＰＣシステム１００内の制御スクリプトを開発するためのフレームワークを提供するものである。ベースライン制御スクリプトはライブラリに格納された共有ベースラインコンポーネントを利用する。図示の実施形態では、共有ベースラインコンポーネントは、制御アルゴリズムを定義するための制御ベースラインライブラリ２４０と、共通に使われる数学関数（ｓｕｍ、ｍｅａｎ、ｍｅｄｉａｎなど）を定義するための数学ベースラインライブラリ２５０と、スクリプト実行の通信アスペクト（マシンインタフェース１９５、オペレータインタフェース２００、他のそのような外部コンポーネントを介してのデータストア２３０、２３２、マシンインタフェース１９５、機器インタフェース１６０との対話）を定義するための対話ベースラインライブラリ２６０と、共有の共通関数を定義するためのユーティリティベースラインライブラリ２７０と、施設に特有の他のライブラリ２４０、２５０、２６０、２７０のルーチンに対する例外または関数を定義するための施設ライブラリ２８０と、ベースライン制御スクリプトに対する呼出に含まれる特定のオペレーションＩＤからのレイヤー（例、ポリゲート層）を定義するためのレイヤーライブラリ２９０とを含む。ベースラインライブラリ２４０、２５０、２６０、２７０、２８０、２９０を、ベースライン制御スクリプト１５２、１５４のオペレーションの間にコントロールエグゼキュータ２２０によってリンクさせておいてもよい。

【００３２】

総じて言うと、ベースライン制御スクリプト１５２、１５４は、スクリプトに対する呼出

に含まれる情報と施設ベースラインライブラリ２８０およびレイヤーベースラインライブラリ２９０に含まれる情報とに基づいてコントロールアクションの性質を判定する。ベースライン制御スクリプト１５２、１５４は、制御ベースラインライブラリ２４０とリンクして必要な制御関数にアクセスする。ベースライン制御スクリプト１５２、１５４は対話ベースラインライブラリ２６０とリンクし、コントロールアクションを実施したりツール１２０、１２５の動作レシピの更新目的で機器インタフェース１６０と通信したりするのに使われるデータを集めるための関数にアクセスする。数学ベースラインライブラリ２５０の関数を必要に応じてベースライン制御スクリプト１５２、１５４または他のライブラリの他の関数から呼ぶようにしてもよい。

【００３３】

ここで図３を参照すると、ベースラインプロセススクリプト１５２の構成を示す簡略ブロック図が示されている。ベースラインプロセススクリプト１５２は、アプリケーションコンフィギュレーションブロック３００と、ベースラインアプリケーションセットアップブロック３１０と、コントローラ定数およびコンテキスト固有の設定ブロック３２０と、フィードフォワードデータ分析ブロック３３０と、制御スレッドブロック３４０と、危機（ｊｅｏｐａｒｄｙ）ブロック３５０と、コントロールアクションおよびビジネスルールブロック３６０と、結果ブロック３７０とを含む。

【００３４】

アプリケーションコンフィギュレーションブロック３００内には、機器インタフェース１６０からの呼出に含まれる情報に基づいてコントローラが使うユーザグローバルコンフィギュレーション変数が定義されている。これは、レシピ管理システム（ＲＭＳ）からの変数の値（すなわちレシピ設定のグローバルデータベース）と必要なコンテキスト変数とを含む。コンテキスト変数の値は制御スレッドを定義し、一般にツール識別コード、ロット番号、オペレーション番号などの変数の値で構成されている。また、必要なベースライン変数も一定の値である。一例として、エラー通知用の電子メール一覧、タイムアウトの値、「子」ロットとみなされる１ロットで許容される最大ウエハ数、コントローラが使った前のレイヤ情報（フィードフォワード情報）などがあげられる。

【００３５】

ベースラインアプリケーションセットアップブロック３１０は、アプリケーションコンフィギュレーションブロック３００でのセット時にロット番号とウエハ数量の値を利用し、ロット番号、ファミリ名、親名、施設、ウエハ数、ステータス（すなわちロットが親ロットであるか子ロットであるか）の値を返す。また、ベースラインアプリケーションセットアップブロック３１０は、コントローラからのポップアップウィンドウとすべてのポップアップウィンドウタイトルの最初の部分の送り先である端末のリストを設定する。

【００３６】

コントローラ定数およびコンテキスト固有の設定ブロック３２０はすでに定義されたコンテキストとＲＭＳ情報とを利用して、制御の動きを計算するのにコントローラが使う値を設定する。たとえば、コントローラ定数およびコンテキスト固有の設定ブロック３２０は、ＲＭＳに定義された値に基づいて、コンテキスト情報（または「スレッド」指定）を使って制御モデルパラメータの値を設定することができる。その具体的な一例は、特定のエッチチャンバのコンテキストに基づいて制御モデルで使われるエッチレートの値や、ＲＭＳに定義されているようなエッチチャンバのエッチレートの値を設定することであろう。また、コントローラ定数およびコンテキスト固有の設定ブロック３２０は、アプリケーションコンフィギュレーションブロック３００に設定されているようなロット番号およびレイヤ名ごとのクエリを使ってデータベースからフィードフォワード情報を読み出す。

【００３７】

フィードフォワードデータ分析ブロック３３０は、特定のロットに関連したデータのアレイに含まれるエレメントをチェックし、抜けている値にはデフォルト値を埋める。たとえば、前のプロセスのターゲットを使って、コントローラが使うフィードフォワード情報の一部として必要な抜けている測定値を設定することができる。デフォルト値を使わずに抜

けているフィードフォワード情報の値を設定するための他の方法をフィードフォワードデータ分析ブロック３３０で実施するようにしてもよい。

【００３８】

制御スレッドブロック３４０は、データストア２３０、２３２を照会して現在の制御スレッドに関連した制御状態を読み出すのに必要なキーと状態構造の値を設定する。これらのキーはデータストア２３０、２３２からスレッド状態データを読み出すのに使われる。制御スレッドブロック３４０は、このスレッドコンテキストで処理された新しいロットの順に並べられたデータのスタックの中のスレッド状態データを検索する。このような値が見つかった場合、これは制御モデルを含むユーザ定義関数に渡され、この関数でスレッド状態の値を計算して返す。スタックに値が見つからなかった場合、制御スレッドブロック３４０は階層を上がって検索し、スレッド状態の値のある最初の階層レベルからデータを読み出す。スタックおよびすべての階層レベルは同様ではあるが精度の異なるデータを含むと仮定される。

【００３９】

危機ブロック３５０は、データベース内でルックアップを実行し、危機スタックのロット数（すなわち、最後の計測オペレーション以降に特定のスレッドで処理されたロットのスタック）についての値を読み出す。この値はこの危機カテゴリ内のロット数についての閾値すなわち一般にＲＭＳで指定される値と比較される。この閾値以下である場合、コントローラはそのまま残る。閾値を超えた場合、危機スタックのロットのリストからのひとつのロットについて計測イベントを実施するようオペレータに指示するポップアップディスプレイを表示してコントローラは異常終了する。

【００４０】

コントロールアクションおよびビジネスルールブロック３６０はコントローラの核心である。コントロールアクションおよびビジネスルールブロック３６０は状態およびターゲット情報からコントローラ入力（プロセスレシピ更新）を計算する。これらの結果はグローバル制御結果アレイに置かれる。次に、コントロールアクションおよびビジネスルールブロック３６０はビジネスルールを実行し、コントローラのユーザ入力オーバーライドに従ってプロセスレシピ更新のチェッキングを制限および／またはプロセスレシピ更新を設定する。

【００４１】

結果ブロック３７０は、プロセスレシピ更新、データ計算／フォーマットを含むコントロールアクションおよびビジネスルールブロック３６０からの出力あるいはイベントを受け、これをまとめて一時的に蓄積し、データをフォーマットする。結果ブロック３７０は一時的に蓄積したデータを機器インタフェース１６０に送信し、機器インタフェース１６０に対してマシンインタフェース１９５によってセットアップ／スタートマシン呼出を開始する。次に、結果ブロック３７０は、現在のコンテキスト（スレッド）についてロット番号およびレイヤに対してデータストア２３０、２３２に格納されたデータを格納する。危機スタックも最後の計測イベント以降に処理された追加のロットとして現在のロットで更新される。

【００４２】

図４を参照すると、ベースライン計測スクリプト１５４の構成を示す簡略ブロック図が示されている。ベースライン計測スクリプト１５４は、計測ツールセットアップブロック４００と、アプリケーションコンフィギュレーションブロック４１０と、ベースラインアプリケーションセットアップブロック４２０と、受入れツールデータブロック４３０と、コントローラ定数およびコンテキスト固有の設定ブロック４４０と、制御スレッドブロック４５０と、モデル更新ブロック４６０と、結果ブロック４７０とを含む。

【００４３】

計測ツールセットアップブロック４００内では、データ収集を開始して一時的に蓄積されていたデータをコントロールエグゼキュータ２２０に送信する。機器インタフェース１６５へのマシン呼出をセットアップ／開始するためにマシンインタフェース１９０も開始さ

れる。

【００４４】

アプリケーションコンフィギュレーションブロック４１０およびベースラインアプリケーションセットアップブロック４２０は、ベースラインプロセススクリプト１５２に関連して上述した同じ名前のブロックと同様の機能を果たす。

【００４５】

受入れツールデータブロック４３０は、ベースライン計測スクリプト１５４を一時停止し、一般に計測ツールであるデータソースからのデータセットを待つ。このイベントの待ち時間とスクリプトの一時停止を解除するイベントの名前が受入れツールデータブロック４３０に指定されている。

【００４６】

コントローラ定数およびコンテキスト固有の設定ブロック４４０は、ベースラインプロセススクリプト１５２に関連して上述した同じ名前のブロックと同様の機能も果たす。

【００４７】

制御スレッドブロック４５０は、現在のスレッドについて算出した制御状態をデータストア２３０、２３２に格納するのに必要なキーおよび状態構造の値を設定する。また、制御スレッドブロック４５０は、スレッド状態を更新するのに必要なすべての値を計算する。この関数は、定義されたグローバル変数を読み取り、必要とされる結果を計算する。これらの結果には、ロット平均、プロセスレート、ターゲットまたは予測からの偏差など、コントローラの更新時に使用される統計値または値が含まれる。この関数の結果はグローバル制御結果アレイにおかれる。

【００４８】

モデル更新ブロック４６０は、ビジネスルール、スペックリミットチェッキング、コントローラのオーバーライドを実施するのに使われる。この関数は、定義されたグローバル変数を読み取り、最終結果を設定する。モデル更新ブロック４６０は、コントローラを更新するのに使用される値ならびに制御履歴に記録される値の設定を担っている。この関数の結果はグローバル制御結果アレイにおかれる。

【００４９】

結果ブロック４７０は、コントローラ定数およびコンテキスト固有の設定ブロック４４０からの出力を受け、これを一時的に蓄積し、機器インタフェース１６５と互換性を持つようにデータをフォーマットする。ベースライン計測スクリプト１５４によるデータ出力も制御履歴ファイルに書き込まれる。供給された変数名に基づいて制御履歴のヘッダが生成される。ログファイルにはファイルの１行目でエンコードされたヘッダがある。計算されたヘッダがファイルの１行目と整合しない場合、既存のファイルの名前を変え、新しいものを開始する。

【００５０】

ここで図５を参照すると、本発明のもうひとつの実施形態による、単一処理ツール１２０上に複数のコントロールアクションを実装できる複数コントローラ用ベースラインプロセススクリプト５００の簡略ブロック図が示されている。たとえば、フォトリソグラフィ用ステッパにオーバーレイコントローラと臨界寸法コントローラの両方を持たせることができる。これらのコントローラは、処理済みウエハからのフィードバックを利用して、露光線量、露光時間、焦点などのさまざまなステッパパラメータを調節する。ポリシリコン層を形成するためのツールなどの成膜ツールにもポリシリコンの粒度やポリシリコン層の厚さなどのパラメータを制御するためのコントローラを複数持たせることができる。

【００５１】

ベースラインプロセススクリプト５００が呼ばれると、このスクリプトは呼出に含まれる情報に基づいて必要なコントロールアクションを判定する。ロットが処理されるコンテキストは、動作対象のコントローラがどれになるかを決める。このコンテキストは、オペレーションＩＤ、エンティティＩＤ、製品ＩＤ、個々の実行（ラン）についての要件を決める他の別個の識別子によって定義される。まず、エンティティＩＤを使ってツールのタイ

プの大雑把なクラスを求める（ステッパ、エッチャー、炉など）。たとえば、エンティティＩＤが処理ツール１２０をステッパであると識別した場合、ステッパ制御コードが呼出される。

【００５２】

ステッパ制御コードの中で、スクリプト内のコンテキスト変数がチェックされ、どの個別コントローラが呼ばれるかを判定する。オペレーションＩＤは、プロセスが走ることを示している（ポリゲートマスクｖｓ第２の層間誘電体層マスク（ＩＬＤ）など）。各コントローラはコンテキスト状況のセットに適用され、これらのコンテキスト条件がすべて満たされた場合にのみ走る。たとえば、ＣＤコントローラは、ポリゲートマスク用に走ることができるが、第２のＩＬＤマスクプロセスでは走ることはできない。一方、オーバーレイコントローラは、両方のマスクイベントで走ることができる。

【００５３】

ベースラインプロセススクリプト５００は、ツールセット（ステッパなど）に基づいて必要なツールコードを整合させるための柔軟性を提供し、利用できるコントローラ（オーバーレイ、ＣＤなど）をすべて走らせる準備をする。同じメインスクリプトを走らせ、同じサブルーチンを呼ぶことが可能であるが、現在のコンテキストで必要なコントローラのみが起動される。

【００５４】

複数コントローラ用ベースラインプロセススクリプト５００は、アプリケーションコンフィギュレーションブロック５１０と、ベースラインアプリケーションセットアップブロック５２０と、コントローラ定数およびコンテキスト固有の設定ブロック５３０と、フィードフォワードデータ分析ブロック５４０と、制御スレッドブロック５５０と、危機ブロック５６０と、コントロールアクションおよびビジネスルールブロック５７０と、結果ブロック５８０とを含む。複数コントローラ用ベースラインプロセススクリプト５００は、後述する点を除いてベースラインプロセススクリプト１５２と同じように動作する。

【００５５】

コントローラ定数およびコンテキスト固有の設定ブロック５３０は、どのコントローラを適用可能であるか（コントローラＡ、コントローラＢあるいは両方など）を判定し、過去に定義されたコンテキストおよびＲＭＳ情報を利用して、制御の動きを算出するのに各コントローラが使う値を設定する。また、コントローラ定数およびコンテキスト固有の設定ブロック５３０は、アプリケーションコンフィギュレーションブロック５１０に設定されているようなロット番号およびレイヤ名ごとのクエリを使って必要なコントローラ各々のフィードフォワード情報をデータベースから読み出す。フィードフォワードデータ分析ブロック５４０は、特定のロットに関連したデータのアレイ中のエレメントをチェックし、各コントローラについて抜けている値のデフォルト値を埋める。

【００５６】

制御スレッドブロック５５０は、データストア２３０、２３２を照会してアクティブなコントローラ各々の現在の制御スレッドに関連した制御状態を読み出すのに必要なキーと状態構造の値を設定する。これらのキーはデータストア２３０、２３２からスレッド状態データを読み出すのに使われる。制御スレッドブロック５５０は、このスレッドコンテキストで処理された新しいロットの順に並べられたデータのスタックのスレッド状態データを検索する。このような値が見つかった場合、これは制御モデルを含むユーザ定義関数に渡され、この関数でスレッド状態の値を計算して返す。スタックに値が見つからなかった場合、制御スレッドブロック３４０は階層を上がって検索し、スレッド状態の値のある最初の階層レベルからデータを読み出す。スタックおよびすべての階層レベルは同様ではあるが精度の異なるデータを含むと仮定される。

【００５７】

コントロールアクションおよびビジネスルールブロック５７０は、各コントローラの状態およびターゲット情報からコントローラ入力（プロセスレシピ更新）を計算する。使用されるコントローラは複数あるため、ひとつのコントローラがそのコントロールアクション

を判定する上で頼りにしている状態情報に他のコントローラが影響を与える場合がある。したがって、それぞれのコントロールアクションを判定する順序を決めるための相対的な優先値をコントローラに割り当てるようにしてもよい。２番目のコントローラの状態情報データを、優先度の高いコントローラがそのコントロールアクション決定に基づいて更新することができる。次に修正された状態情報に基づいて２番目のコントローラがそのコントロールアクションを判定する。このようにして連携することで、コントローラは動作レシピの変更に関して互いに競合することがなくなる。

【００５８】

結果ブロック５８０は、アクティブなコントローラすべてからコントロールアクション出力を集め、このデータを一時的に蓄積し、データをフォーマットする。結果ブロック５８０は一時的に蓄積したデータを機器インタフェース１６０に送信し、機器インタフェース１６０に対してマシンインタフェース１９５によってセットアップ／スタートマシン呼出を開始する。次に、結果ブロック５８０は、現在のコンテキスト（スレッド）についてロット番号およびレイヤに対してデータストア２３０、２３２に格納されたデータを格納し、危機スタックを更新する。

【００５９】

ここで図６を参照すると、本発明の他の実施形態による複数のコントローラを統合するための方法の簡略流れ図が示されている。ブロック６００では、複数のツールで被加工物を処理する。ブロック６１０では、（制御実行マネージャ１５０によるなどの方法で）複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始する。ベースライン制御スクリプトの開始後、コントロールエグゼキュータ２２０が残りのタスクを実施する。ブロック６２０では選択したツールに必要な制御ルーチン群を特定する。ブロック６３０では、必要な制御ルーチン群について選択したツールに関連した過去のコントロールアクションに関する制御状態情報を読み出す。ブロック６４０では、必要な制御ルーチン群からの最初の制御ルーチンを実行し、最初のコントロールアクションを生成する。ブロック６５０では、最初のコントロールアクションに基づいて必要な制御ルーチン群からの２番目の制御ルーチンに関連した制御状態情報を変更する。ブロック６６０では、修正後の制御状態情報に基づいて２番目の制御ルーチンを実行し、２番目のコントロールアクションを生成する。

【００６０】

本発明は本願明細書の教示内容の利益を享受する当業者らに自明の上記とは異なるが等価な方法で改変および実施することのできるものであるため、上記にて開示した個々の実施形態は一例にすぎない。さらに、添付の請求の範囲に記載したものを除き、本願明細書に図示した構造または設計の詳細に対する限定を何ら意図するものではない。したがって、上記にて開示した個々の実施形態を変更または改変してもよく、そのような変形例はいずれも本発明の範囲に包含されるとみなせることは明白である。よって、本願明細書で求める保護は添付の請求の範囲に記載のとおりである。

【図面の簡単な説明】

【００６１】

【図１】本発明の一実施形態による高度プロセス制御（ＡＰＣ）システムの簡略ブロック図である。

【図２】図１のシステム内のベースライン制御スクリプトと、多様な共有ベースラインライブラリとの間のリンクを示す図である。

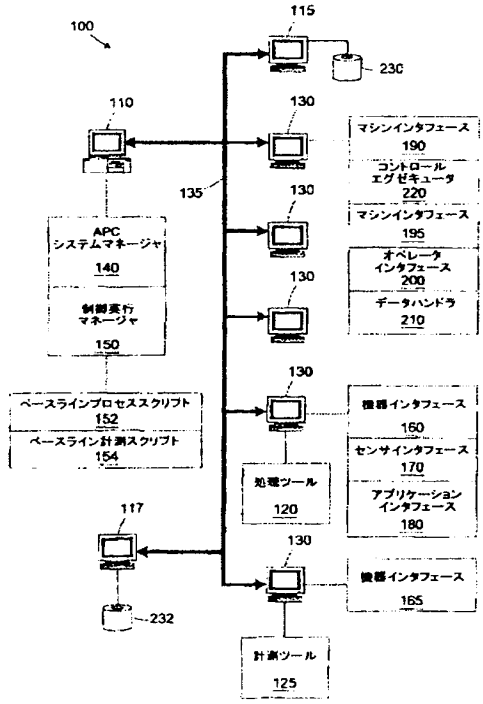
【図３】ベースラインプロセススクリプトの構成を示す簡略ブロック図である。

【図４】ベースライン計測スクリプトの構成を示す簡略ブロック図である。

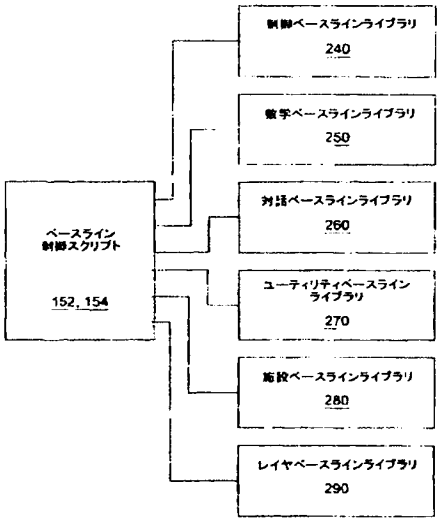
【図５】複数コントローラ用ベースラインプロセススクリプトの構成を示す簡略ブロック図である。

【図６】本発明のもうひとつの実施形態による複数のコントローラを統合するための方法の簡略流れ図である。

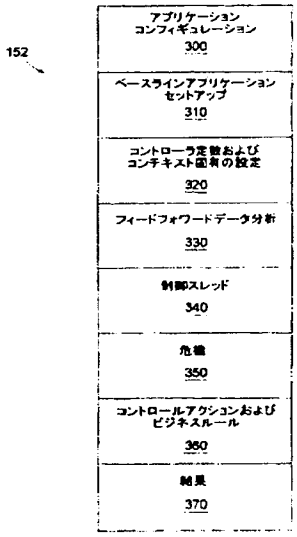
【図１】


100
115
230
110
130
マシンインタフェース
190
コントロール
エグゼキュータ
220
135
130
マシンインタフェース
195
APC
システムマネージャ
140
オペレータ
インタフェース
200
130
制御実行
マネージャ
150
データハンドラ
210
130
ベースラインプロセススクリプト
152
機器インタフェース
160
ベースライン計測スクリプト
154
センサインタフェース
170
処理ツール
120
アプリケーション
インタフェース
180
117
130
機器インタフェース
165
232
計測ツール
125

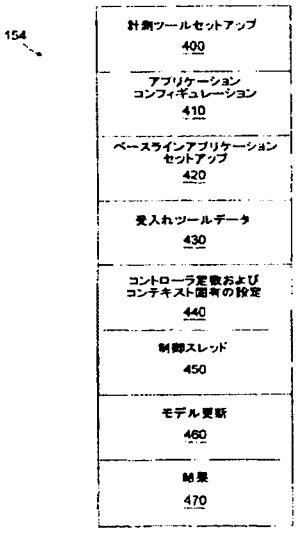
【図２】


制御ベースラインライブラリ
240
数学ベースラインライブラリ
250
ベースライン
制御スクリプト
152, 154
対話ベースラインライブラリ
260
ユーティリティベースライン
ライブラリ
270
施設ベースラインライブラリ
280
レイヤベースラインライブラリ
290

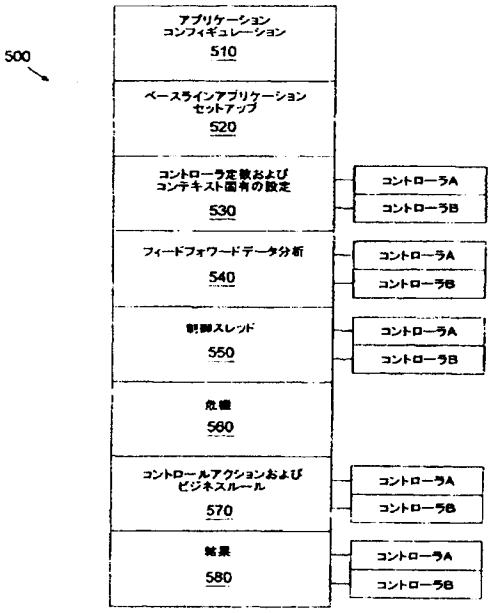
【図３】


152
アプリケーション
コンフィギュレーション
300
ベースラインアプリケーション
セットアップ
310
コントローラ定数および
コンテキスト固有の設定
320
フィードフォワードデータ分析
330
制御スレッド
340
危機
350
コントロールアクションおよび
ビジネスルール
360
結果
370

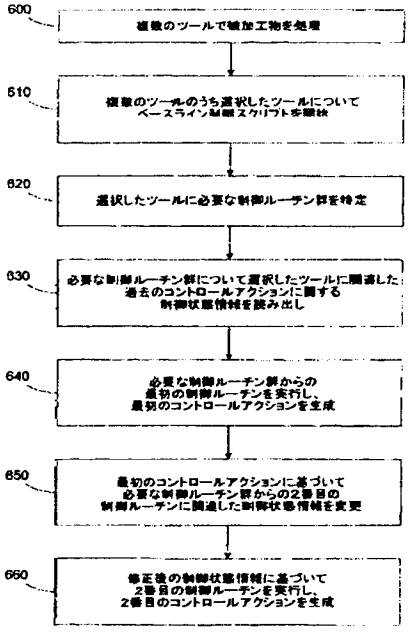
【図４】


154
計測ツールセットアップ
400
アプリケーション
コンフィギュレーション
410
ベースラインアプリケーション
セットアップ
420
受入れツールデータ
430
コントローラ定数および
コンテキスト固有の設定
440
制御スレッド
450
モデル更新
460
結果
470

【図５】


500
アプリケーション
コンフィギュレーション
510
ベースラインアプリケーション
セットアップ
520
コントローラ定義および
コンテキスト固有の設定
530
フィードフォワードデータ分析
540
制御スレッド
550
危機
560
コントロールアクションおよび
ビジネスルール
570
結果
580
コントローラA
コントローラB

【図６】


600
複数のツールで被加工物を処理
610
複数のツールのうち選択したツールについて
ベースライン制御スクリプト手順秋
620
選択したツールに必要な制御ルーチン群を特定
630
必要な制御ルーチン群について選択したツールに関連した
過去のコントロールアクションに関する
制御状態情報を読み出し
640
必要な制御ルーチン群からの
最初の制御ルーチンを実行し、
最初のコントロールアクションを生成
650
最初のコントロールアクションに基づいて
必要な制御ルーチン群からの2番目の
制御ルーチンに関連した制御状態情報を変更
660
修正後の制御状態情報に基づいて
2番目の制御ルーチンを実行し、
2番目のコントロールアクションを生成

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069063 A2

(51) International Patent Classification[7]: G05B 19/418, 19/042

(21) International Application Number: PCT/US01/50178

(22) International Filing Date: 22 October 2001 (22.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/789,871 21 February 2001 (21.02.2001) US

(71) Applicant: **ADVANCED MICRO DEVICES, INC.** [US/US]; One AMD Place, Mail Stop 68, Sunnyvale, CA 94088-3453 (US).

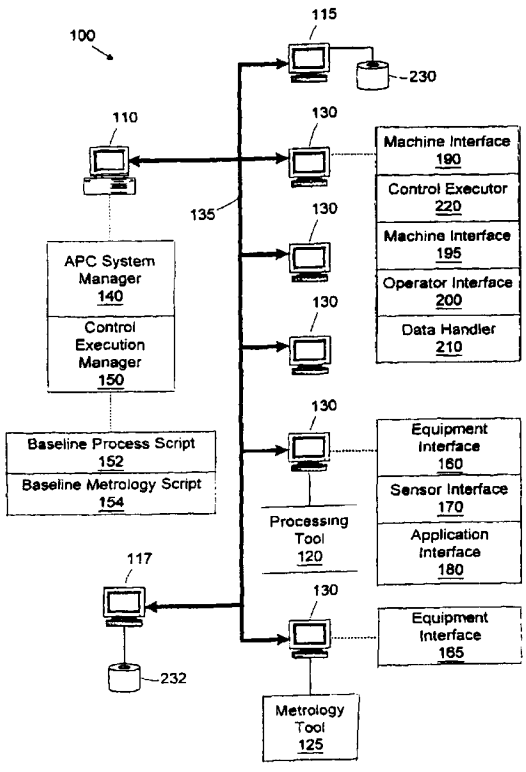
(72) Inventors: **BODE, Christopher, A.**; 4700 Staggerbrush Road #738, Austin, TX 78749 (US). **PASADYN, Alexander, J.**; 8717 Dandelion Trail, Austin, TX 78745 (US). **TOPRAC, Anthony, J.**; 4023 Walnut Clay, Austin, TX 78731 (US). **HEWETT, Joyce, S., Oey**; 6517 Sans Souci Cove, Austin, TX 78759 (US). **PETERSON, Anastasia, Oshelski**; 5811 Cannon Mountain Drive, Austin, TX 78749 (US). **SONDERMAN, Thomas, J.**; 16010 Braesgate Drive, Austin, TX 78717 (US). **MILLER, Michael, L.**; 2614 Little Elm Trail, Cedar Park, TX 78613 (US).

(74) Agent: **DRAKE, Paul, S.**; Advanced Micro Devices, Inc., 5204 East Ben White Boulevard, Mail Stop 562, Austin, TX 78741 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR CONTROLLING A TOOL USING A BASELINE CONTROL SCRIPT



(57) Abstract: A method for controlling a manufacturing system (100) includes processing workpieces in a plurality of tools (120, 125); initiating a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125); providing context information for the baseline control script (152, 154); determining a tool type based on the context information; selecting a control routine for the selected tool (120, 125) based on the tool type; and executing the control routine to generate a control action for the selected tool (120, 125). A manufacturing system (100) includes a plurality of tools (120, 125) adapted to process workpieces, a control execution manager (150), and a control executor (220). The control execution manager (150) is adapted to initiate a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125) and provide context information for the baseline control script (152, 154). The control executor (220) is adapted to execute the baseline control script (152, 154), determine a tool type based on the context information, select a control routine for the selected tool (120, 125) based on the tool type, and execute the control routine to generate a control action for the selected tool (120, 125).

WO 02/069063 A2

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND APPARATUS FOR CONTROLLING A TOOL USING A BASELINE CONTROL SCRIPT

TECHNICAL FIELD

This invention relates generally to the field of semiconductor device manufacturing and, more particularly, to a method and apparatus for controlling a tool using a baseline control script.

BACKGROUND ART

There is a constant drive within the semiconductor industry to increase the quality, reliability and throughput of integrated circuit devices, *e.g.*, microprocessors, memory devices, and the like. This drive is fueled by consumer demands for higher quality computers and electronic devices that operate more reliably. These demands have resulted in a continual improvement in the manufacture of semiconductor devices, *e.g.*, transistors, as well as in the manufacture of integrated circuit devices incorporating such transistors. Additionally, reducing the defects in the manufacture of the components of a typical transistor also lowers the overall cost per transistor as well as the cost of integrated circuit devices incorporating such transistors.

Generally, a set of processing steps is performed on a lot of wafers using a variety of processing tools, including photolithography steppers, etch tools, deposition tools, polishing tools, rapid thermal processing tools, implantation tools, *etc*. The technologies underlying semiconductor processing tools have attracted increased attention over the last several years, resulting in substantial refinements. However, despite the advances made in this area, many of the processing tools that are currently commercially available suffer certain deficiencies. In particular, such tools often lack advanced process data monitoring capabilities, such as the ability to provide historical parametric data in a user-friendly format, as well as event logging, real-time graphical display of both current processing parameters and the processing parameters of the entire run, and remote, *i.e.*, local site and worldwide, monitoring. These deficiencies can engender nonoptimal control of critical processing parameters, such as throughput, accuracy, stability and repeatability, processing temperatures, mechanical tool parameters, and the like. This variability manifests itself as within-run disparities, run-to-run disparities and tool-to-tool disparities that can propagate into deviations in product quality and performance, whereas an ideal monitoring and diagnostics system for such tools would provide a means of monitoring this variability, as well as providing means for optimizing control of critical parameters.

One technique for improving the operation of a semiconductor processing line includes using a factory wide control system to automatically control the operation of the various processing tools. The manufacturing tools communicate with a manufacturing framework or a network of processing modules. Each manufacturing tool is generally connected to an equipment interface. The equipment interface is connected to a machine interface which facilitates communications between the manufacturing tool and the manufacturing framework. The machine interface can generally be part of an advanced process control (APC) system. The APC system initiates a control script based upon a manufacturing model, which can be a software program that automatically retrieves the data needed to execute a manufacturing process. Often, semiconductor devices are staged through multiple manufacturing tools for multiple processes, generating data relating to the quality of the processed semiconductor devices.

During the fabrication process, various events may take place that affect the performance of the devices being fabricated. That is, variations in the fabrication process steps result in device performance variations.

Factors, such as feature critical dimensions, doping levels, contact resistance, particle contamination, *etc.*, all may potentially affect the end performance of the device. Various tools in the processing line are controlled in accordance with performance models to reduce processing variation. Commonly controlled tools include photolithography steppers, polishing tools, etching tools, and deposition tools. Pre-processing and/or post-processing metrology data is supplied to process controllers for the tools. Operating recipe parameters, such as processing time, are calculated by the process controllers based on the performance model and the metrology information to attempt to achieve post-processing results as close to a target value as possible. Reducing variation in this manner leads to increased throughput, reduced cost, higher device performance, *etc.*, all of which equate to increased profitability.

Configuration control and efficiency issues are prevalent in a distributed computing environment, such as a factory-wide APC system. Typically, there are numerous software developers writing control code to construct the process controllers. One particular developer may work extensively developing controllers of a certain type. It is common for each developer to have a unique programming style, and to rely on routines that they have created themselves. For example, each developer may have a set of routines for interfacing with databases or other entities within the APC framework and for performing various math functions and basic utility functions.

One problem associated with such an arrangement is that there is little consistency between process control scripts. The large number of custom scripts also presents a configuration control problem and an efficiency problem. Developers may spend considerable time duplicating code that has already been developed, perhaps for a different type of process controller that a different developer has created. Debugging non-standardized code is also more time-consuming and further reduces efficiency.

The present invention is directed to overcoming, or at least reducing the effects of, one or more of the problems set forth above.

DISCLOSURE OF INVENTION

One aspect of the present invention is seen in a method for controlling a manufacturing system. The method includes processing workpieces in a plurality of tools; initiating a baseline control script for a selected tool of the plurality of tools; providing context information for the baseline control script; determining a tool type based on the context information; selecting a control routine for the selected tool based on the tool type; and executing the control routine to generate a control action for the selected tool.

Another aspect of the present invention is seen in a manufacturing system including a plurality of tools adapted to process workpieces, a control execution manager, and a control executor. The control execution manager is adapted to initiate a baseline control script for a selected tool of the plurality of tools and provide context information for the baseline control script. The control executor is adapted to execute the baseline control script, determine a tool type based on the context information, select a control routine for the selected tool based on the tool type, and execute the control routine to generate a control action for the selected tool.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention may be understood by reference to the following description taken in conjunction with the accompanying drawings, in which like reference numerals identify like elements, and in which:

Figure 1 is a simplified block diagram of an advanced process control (APC) system in accordance with one illustrative embodiment of the present invention;

Figure 2 is a diagram illustrating links between a baseline control scripts in the system of Figure 1 and a variety of shared baseline libraries;

Figure 3 is a simplified block diagram illustrating the organization of a baseline process script;

Figure 4 is a simplified block diagram illustrating the organization of a baseline metrology script;

Figure 5 is a simplified block diagram illustrating the organization of a multiple controller baseline process script; and

Figure 6 is a simplified flow diagram of a method for integrating multiple controllers in accordance with another illustrative embodiment of the present invention.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and are herein described in detail. It should be understood, however, that the description herein of specific embodiments is not intended to limit the invention to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the scope of the invention as defined by the appended claims.

MODE(S) FOR CARRYING OUT THE INVENTION

Illustrative embodiments of the invention are described below. In the interest of clarity, not all features of an actual implementation are described in this specification. It will of course be appreciated that in the development of any such actual embodiment, numerous implementation-specific decisions must be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, which will vary from one implementation to another. Moreover, it will be appreciated that such a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking for those of ordinary skill in the art having the benefit of this disclosure.

Referring now to the figures, and first to Figure 1, a simplified block diagram of an advanced process control (APC) system 100 is shown. The APC System 100 is a distributed software system of interchangeable, standardized software components permitting run-to-run control and fault detection/classification. The software components implement an architectural standard based on a Semiconductor Equipment and Materials International (SEMI) Computer Integrated Manufacturing (CIM) Framework compliant system technologies specification and an Advanced Process Control (APC) Framework. CIM (SEMI E81-0699 - Provisional Specification for CIM Framework Domain Architecture) and APC (SEMI E93-0999 - Provisional Specification for CIM Framework Advanced Process Control Component) specifications are publicly available from SEMI. This particular architecture relies heavily on software utilizing object oriented programming and employs the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) and CORBA_Services specifications for distributed object systems. Information and specifications for the OMG CORBA architecture are also readily, publicly available. An exemplary software system capable of being adapted to perform the functions of the APC system 100 as described herein is the Catalyst system offered by KLA-Tencor, Inc.

The components communicate with each other using the CORBA Interface Definition Language (IDL) and rely on a common set of services to support their interaction. A standard set of distributed-object services are defined by the OMG. Among these services are:

CORBA - the standard-based communications protocol used for all direct component-to-component interaction. Standard interfaces can be defined according to an object-oriented, remote invocation

communications model. These interfaces and all APC communications are defined using IDL. Components communicate by invoking operations on each others interfaces. Data is passed between components as operation parameters and return values.

OMG Event Service - supports asynchronous communications between components. Many of the APC objects emit events as they change state. These events are received by interested event subscribers. Examples of event usage within the APC system include, but are not limited to, communication component state (including error state), notification of fault alarms detected by fault detection and classification software, and reporting of machine status and collected data.

OMG Trading Service - enables a component to find another component with which to interact. When a component is installed, a description of its services (a services offer) is exported to the Trading Service. Another component can later request a list of service providers that meet certain criteria. The Trading Service supplies a list of other components that can provide the requested service. That capability is used upon component startup to allow one component to find other components with which it must interact. It is also used upon Plan Startup when a Plan Execution component needs to find Capability Providers to provide the required capabilities specified in the plan.

These services are well known in the art. OMG's CORBA/IIOP Specifications document and CORBA Services Specification documents are widely distributed among those in the art and provide greater detail.

In the illustrated embodiment, the APC system 100 is adapted to control a semiconductor manufacturing environment. The components communicate with each other using CORBA Interface Definition Language (IDL). The cooperating software components manage process control plans/strategies; collect data from process equipment, metrology tools, and add-on sensors; invoke various process control applications/algorithms with this information; and update process models and modify/download tool operating recipe parameters as appropriate. The APC system 100 is a factory-wide software system for controlling semiconductor production processes, but this is not necessary to the practice of the invention. The strategies taught by the present invention can be applied to different computer systems, on any scale.

In an exemplary implementation, the APC system 100 includes an APC host computer 110, database servers 115, 117, a processing tool 120, a metrology tool 125, and one or more workstations 130. The components of the APC system are interconnected through a bus 135. The bus 135 may actually include multiple layers and use multiple protocols. Overall operation of the APC system 100 is directed by an APC system manager 140 resident on an APC host computer 110. The APC system manager 140 provides administrative, configuration, event, and state services for all servers developed for the APC Framework; definition, grouping, installation, and management of the components in the APC system 100; centralized services for capturing activity and trace information for diagnostic and monitoring purposes; a centralized repository of component configuration information, including setup values, system environment settings; and lists of dependent objects and event channels. However, in alternative embodiments, these functions may be divided into one or more software components, e.g., a base manager, a system manager, a logger, and a registry.

The APC system 100 includes a network of processing modules. These processing modules are sometimes referred to as "integration components." Integration components serve as interfaces to existing factory systems, and provide capabilities for running APC Plans. An "APC Plan" is an application program called to perform some specific task, as is discussed more fully below. The integration components are shown

as they might be hosted by the various processing resources within the APC system 100. These specific hosting locations are provided for exemplary purposes. The processing resources are interconnected, and the various software components may be either distributed among the various computers or centralized, depending on the complexity of the system. Each of the integration components in this particular embodiment, are software-implemented. They are programmed in C++ using object-oriented programming techniques as are known in the art. An advantage of the APC system 100 is its modular structure, which provides portability of software components. The integration components include, but are not limited to, the APC system manager 140; a control execution manager 150; equipment interfaces 160, 165 associated with the tools 120, 125; a sensor interface 170 associated with the processing tool 120; an application interface 180; machine interfaces 190, 195; an operator interface 200; and a data handler 210.

The control execution manager 150 is the component primarily responsible for "choreographing" the operation of the APC System 100. The control execution manager 150 interprets APC plans, executes main scripts and subscripts, and invokes event scripts as events dictate. A variety of plans, scripts, and subscripts may be used in various implementations. The specific number and function of various plans, scripts, and subscripts will be implementation specific. For instance, the present embodiment includes, but is not limited to, the following plans:

a data collection plan - a data structure used by sensor and machine interfaces defining the requirements for what data should be collected from a specific processing equipment, and how that data should be reported back;

a duration plan - a plan that defines trigger conditions and trigger delays that cause sensors to act, e.g., start data collection, stop data collection;

a reporting plan - a plan that defines what to do with the collected data, as well as when to signal the data's availability; and

a sampling plan - a plan that defines the frequency at which the data is to be collected by an external sensor;

a control plan - a collection of control scripts designed to be used together to perform APC activities; and

a control script - a sequence of actions/activities that the APC system is to execute under a particular defined situation.

The control execution manager 150 coordinates the execution of user-defined process control plans among all the integration components for a given tool, such as the processing tool 120. When instructed, the control execution manager 150 retrieves a plan and its associated scripts. It preprocesses subscripts to provide routines to main and event scripts. It also obtains a list of the capabilities necessary to execute the plan, as specified in the plan and connects to the proper integration components providing the required capabilities.

The control execution manager 150 then delegates responsibility to run the plan to a control executor 220. In the illustrated embodiment, the control execution manager 150 uses baseline control scripts for determining control actions to be performed. A baseline process script 152 is designated for use with processing tools, such as the processing tool 120, and a baseline metrology script 154 is designated for use with metrology tools, such as the metrology tool 125. A more detailed discussion of the baseline scripts 152, 154 is provided below in reference to Figures 2 through 6.

The control execution manager 150 creates a control executor 220 based on the appropriate baseline process script 152 or baseline metrology script 154 to sequentially execute the plan and report completion of the plan or errors in the execution of the plan back to the control execution manager 150. Thus, while the control execution manager 150 is responsible for the overall management of all plans executed, each control executor 220 is responsible for running only one plan. The control executor 220 is created by the control execution manager 150, exists for the life of the plan, and is destroyed by the control execution manager 150 after reporting that the plan is completed or aborted. The control execution manager 150 can start multiple plans concurrently via multiple control executors 220.

The machine interfaces 190, 195 bridge the gap between the APC framework, e.g., the APC system manager 140, and the equipment interfaces 160, 165. The machine interfaces 190, 195 interface the processing or metrology tools 120, 125 with the APC framework and support machine setup, activation, monitoring, and data collection. In this particular embodiment, the machine interfaces 190, 195 primarily translate between specific communications of the equipment interfaces 160, 165 and CORBA communications of the APC framework. More particularly, the machine interfaces 190, 195 receive commands, status events, and collected data from the equipment interfaces 160, 165 and forward as needed to other APC components and event channels. In turn, responses from other APC components are received by the machine interfaces 190, 195 and routed to the equipment interfaces 160, 165. The machine interfaces 190, 195 also reformat and restructure messages and data as necessary. The machine interfaces 190, 195 support the startup/shutdown procedures within the APC System Manager 140. They also serve as APC data collectors, buffering data collected by the equipment interfaces 160, 165 and emitting appropriate data collection events.

The sensor interface 170 collects data generated by the sensors monitoring the operation of the processing tool 120. The sensor interface 170 provides the appropriate interface environment to communicate with external sensors, such as LabVIEW or other sensor, bus-based data acquisition software. The application interface 180 provides the appropriate interface environment to execute control plug-in applications such as LabVIEW, Mathematica, ModelWare, MatLab, Simca 4000, and Excel. The sensors may be supplied with the processing tool 120 by the original equipment manufacturer (OEM) or they may be "add-on" sensors installed subsequent to acquisition from the OEM. The sensor interface 170 collects data generated by the sensors. The sensors may generate data on, for instance, the pressure and temperature of the operating conditions. The application interface 180 takes data from the control executor 220 and performs calculations or analysis on that data. The results are then returned to the control executor 220. The machine interface 190 and the sensor interface 170 use a common set of functionality to collect data to be used. The equipment interface 160 gathers the respective data collected by the sensors on the processing tool 120 and transmits the gathered data to the machine interface 190.

The operator interface 200 facilitates communication between a wafer fabrication technician and the APC system 100 via a graphical user interface (GUI) (not shown). The GUI may be a Windows® or UNIX based operating system. However, this is not necessary to the practice of the invention. Indeed, some alternative embodiments might not even employ a GUI and may communicate through a disk operating system (DOS) based operating system. The operator interface 200 displays dialogue boxes to provide information, request guidance and collect additional data. Through a CORBA interface, the operator interface 200 component allows technicians to display a variety of popup dialogs simultaneously on any number of display

groups. The operator interface 200 also maintains a group of displays in which a popup could appear. The operator interface 200 may also provide an announcement operation, i.e., a one-way message that displays a simple popup with message and "OK" button.

The data handler 210 receives data generated by other APC system 100 components and stores the data in data stores 230, 232 (e.g., relational databases) on the database servers 115, 117. The data handler 210 may be adapted to receive standard structured query language (SQL) commands, or alternatively, the data handler 210 may translate a different type of access protocol to generate a SQL command or some other protocol command. Centralizing the data storage functions increases the portability of the various components.

The general operation of the baseline control scripts 152, 154 is described with reference to the simplified block diagram provided in Figure 2, which illustrates the links between the baseline control scripts 152, 154 and a variety of shared baseline libraries. In general, a baseline control script 152, 154 provides a framework for developing control scripts within the APC system 100. The baseline control scripts use shared baseline components stored in libraries. In the illustrated embodiment, the shared baseline components include a control baseline library 240 for defining control algorithms; a math baseline library 250 for defining commonly used math functions (*e.g.*, sum, mean, median, *etc.*); an interaction baseline library 260 for defining communication aspects of script execution (*e.g.*, interaction with the data stores 230, 232, the machine interface 195, the equipment interface 160 through the machine interface 195, the operator interface 200, and other such external components); a utility baseline library 270 for defining shared common functions; a facility library 280 for defining functions or exceptions to the routines in the other libraries 240, 250, 260, 270 that are specific to the facility; and a layer library 290 for defining the layer (ex. poly gate layer) from the particular operation ID included in the call to the baseline control script. The baseline libraries 240, 250, 260, 270, 280, 290 may be linked to during the operation of the baseline control scripts 152, 154 by the control executor 220.

In general, the baseline control script 152, 154 determines the nature of the control action, based on the information included in the call to the script and the information in the facility and layer baseline libraries 280, 290. The baseline control script 152, 154 links to the control baseline library 240 to access the necessary control functions. The baseline control script 152, 154 links to the interaction baseline library 260 to access functions for gathering the data used to perform the control action and to communicating with the equipment interface 160 for updating the operating recipe of the tool 120, 125. The functions in the math baseline library 250 may be called by other functions in the baseline control script 152, 154 or other libraries, as necessary.

Turning now to Figure 3, a simplified block diagram illustrating the organization of the baseline process script 152 is provided. The baseline process script 152 includes an application configuration block 300, a baseline application setup block 310, a controller constants and context-specific settings block 320, a feed forward data analysis block 330, a control thread block 340, a jeopardy block 350, a control action and business rules block 360, and a results block 370.

Within the application configuration block 300, user global configuration variables are defined for use by the controller based on the information contained in the call from the equipment interface 160. This includes the values of variables from the recipe management system (RMS) (*i.e.*, a global database of recipe settings) and the required context variables. Context variable values define the control thread and typically consist of values for variables such as tool identification code, lot number, operation number, and so forth. In addition, any needed baseline variables are also given values. Examples include an email list for error notifications, values for

timeouts, the maximum number of wafers allowed in a lot considered as a "child" lot, previous layer information that the controller uses (feed forward information) and so forth.

The baseline application setup block 310 uses the values for lot number and quantity of wafers, as set in the application configuration block 300 and returns values for lot number, family name, parent name, facility, number of wafers, and status (*i.e.*, whether lot is a parent or a child lot). The baseline application setup block 310 also sets the default list of terminals to which the controller will send popup windows, as well as the first part of all popup window titles.

The controller constants and context-specific settings block 320 uses the previously defined context and RMS information to set the values that the controller uses to calculate control moves. For example, the controller constants and context-specific settings block 320 may use the context information (or "thread" designation) to set the value of a control model parameter according to the value defined in RMS. A specific example would be setting the value for etch rate used in a control model according to the context of the particular etch chamber and the value for that etch chamber's etch rate as defined in RMS. In addition, the controller constants and context-specific settings block 320 retrieves feed forward information from the database using a query by lot number and layer name as set in the application configuration block 300.

The feed forward data analysis block 330 checks the elements in an array of data associated with a given lot and fills in default values for missing values. For example, the target of a previous process may be used to set the value of a missing measurement needed as part of the feed forward information used by the controller. Other methods for setting the value of missing feed forward information, in lieu of using a default value, may also be performed in the feed forward data analysis block 330.

The control thread block 340 sets the values of the keys and state structures needed for querying the data stores 230, 232 to retrieve control states associated with the current control thread. The keys are used to retrieve the thread state data from the data stores 230, 232. The control thread block 340 searches for the thread state data is in the stack of ordered data for recent lots processed with this thread context. If such values are found they are passed to a user-defined function containing the control model, which calculates and returns values for the thread states. If no values in the stack are found, the control thread block 340 searches up the hierarchy and retrieves the data from the first hierarchy level that has values for the thread states. The stack and all hierarchy levels are assumed to contain similar data, but at differing degrees of precision.

The jeopardy block 350 performs a lookup in the database and retrieves the value for the number of lots in the jeopardy stack (*i.e.*, the stack of lots which were processed on the given thread since the last metrology operation). This value is compared to a threshold value for number of lots in this jeopardy category, a value typically specified in RMS. If this threshold value is not exceeded, the controller continues. If the threshold value is exceeded, the controller aborts with a popup display that instructs the operator to perform a metrology event on a lot from the list of lots in the jeopardy stack.

The control action and business rules block 360 is the heart of the controller. The control action and business rules block 360 computes the controller inputs (process recipe updates) from the state and target information. These results are placed in a global control results array. Next, the control action and business rules block 360 executes business rules, limits checking on the process recipe updates, and/or sets the process recipe updates according to user-input overrides of the controller.

The results block 370 takes the output from the control action and business rules block 360, including process recipe updates, data calculations/formatting, or events, buffers it together and formats the data. The results block 370 sends the buffered data to the equipment interface 160 and initiates setup/start machine calls by the machine interface 195 to the equipment interface 160. Next, the results block 370 stores the data is stored in the data stores 230, 232 against the lot number and layer for the current context (thread). The jeopardy stack is also updated with the current lot as an additional lot processed since the last metrology event.

Referring now to Figure 4, a simplified block diagram illustrating the organization of the baseline metrology script 154 is provided. The baseline metrology script 154 includes a metrology tool setup block 400, an application configuration block 410, a baseline application setup block 420, an incoming tool data block 430, a controller constants and context-specific settings block 440, a control thread block 450, a model update block 460, and a results block 470.

Within the metrology tool setup block 400, data collection is initiated and any buffered data is sent to the control executor 220. The machine interface 190 is also initiated to send setup/start machine calls to the equipment interface 165.

The application configuration block 410 and the baseline application setup block 420 perform functions similar to the blocks of the same name described above in relation to the baseline process script 152.

The incoming tool data block 430 pauses the baseline metrology script 154 to wait for a data set from a data source, typically a metrology tool. The time period for waiting for this event and the name of the event which will release the script pause are specified in the incoming tool data block 430.

The controller constants and context-specific settings block 440 also performs functions similar to the block of the same name described above in relation to the baseline process script 152.

The control threads block 450 sets the values of the keys and state structures needed for storing calculated control states for the current thread to the data stores 230, 232. In addition, the control threads block 450 computes all values needed to update the thread states. This function read the defined global variables and computes the needed results. The results include statistics or values that are used in updating the controller, such as lot averages, process rates, and deviations from targets or predictions. The results of this function are placed in the global control results array.

The model update block 460 is used to perform business rules, spec limit checking, and overrides of the controller. This function reads the defined global variables and sets the final results. The model update block 460 is responsible for setting the values that will be used to update the controller as well as the values that will be logged to the control history. The results of this function are placed in the global control results array.

The results block 470 takes the output from the controller constants and context-specific settings block 440, buffers it and formats the data to be compatible with the equipment interface 165. The data output by the baseline metrology script 154 is also written to a control history file. A header for the control history is generated based on the supplied variable names. The log files have the header encoded in the first line of the file. If the computed header does not match the first line of the file, the existing file is renamed, and a new one is started.

Turning now to Figure 5, a simplified block diagram of a multiple controller baseline process script 500 capable of implementing multiple control actions on a single processing tool 120 in accordance with another embodiment of the present invention is provided. For example, a photolithography stepper may have both an

overlay controller and a critical dimension controller. The controllers use feedback from processed wafers to adjust various stepper parameters, such as exposure dose, exposure time, focus, *etc.* A deposition tool, such as a tool for forming polysilicon layers, may also have multiple controllers for controlling parameters such as polysilicon grain size and polysilicon layer thickness.

When the baseline process script 500 is called, it determines the necessary control actions based on the information included in the call. The context in which a lot is to be processed determines which of the controllers that will be run. The context is defined by the operation ID, the entity ID, the product ID, and other such discrete identifiers that determine the requirements for a particular run. First, the entity ID is used to determine the general class of tool type (*e.g.*, stepper, etcher, furnace, *etc.*). For example, if the entity ID identifies the processing tool 120 as a stepper, the stepper control code is called.

Within the stepper control code, the context variables within the script are checked to determine which individual controller are to be called. The operation ID indicates the process will be run (*e.g.*, poly gate mask vs. second interlevel dielectric layer mask(ILD)). Each controller applies to a set of context situations and will only run if all of those context conditions are met. For example, the CD controller may run for a poly gate mask, but may not run at the second ILD mask process. The overlay controller, on the other hand, may run at both mask events.

The baseline process script 500 provides the flexibility to match the required tool code based on the tool set (*e.g.*, steppers) and prepare to run all of the available controllers (*e.g.*, overlay, CD, *etc.*). The same main script is run, the same subroutines are available to be called, but only the controllers needed in the current context are activated.

The multiple controller baseline process script 500 includes an application configuration block 510, a baseline application setup block 520, a controller constants and context-specific settings block 530, a feed forward data analysis block 540, a control thread block 550, a jeopardy block 560, a control action and business rules block 570, and a results block 580. The multiple controller baseline process script 500 operates in a similar manner to the baseline process script 152, except as described below.

The controller constants and context-specific settings block 530 determines which of the controllers is applicable (*e.g.*, controller A, controller B, or both) and uses the previously defined context and RMS information to set the values that each controller uses to calculate control moves. The controller constants and context-specific settings block 530 also retrieves feed forward information for each of the required controllers from the database using a query by lot number and layer name as set in the application configuration block 510. The feed forward data analysis block 540 checks the elements in an array of data associated with a given lot and fills in default values for missing values for each controller.

The control thread block 550 sets the values of the keys and state structures needed for querying the data stores 230, 232 to retrieve control states associated with the current control threads for each of the active controllers. The keys are used to retrieve the thread state data from the data stores 230, 232. The control thread block 550 searches for the thread state data is in the stack of ordered data for recent lots processed with this thread context. If such values are found they are passed to a user-defined function containing the control model, which calculates and returns values for the thread states. If no values in the stack are found, the control thread block 340 searches up the hierarchy and retrieves the data from the first hierarchy level that has values for the

thread states. The stack and all hierarchy levels are assumed to contain similar data, but at differing degrees of precision.

The control action and business rules block 570 computes the controller inputs (process recipe updates) from the state and target information for each of the controllers. Because multiple controllers are used, one controller may affect the state information relied on by the other controller for determining its control action. Hence, the controllers may be assigned relative priority values for determining the order in which their control actions are determined. The higher priority controller may update the state information data for the second controller based on its control action determination. The second controller than determines its control action based on the modified state information. By cooperating in such a manner, the controllers will not compete with each other regarding operating recipe changes.

The results block 580 gathers the control action output from all of the active controllers, buffers the data, and formats the data. The results block 580 sends the buffered data to the equipment interface 160 and initiates setup/start machine calls by the machine interface 195 to the equipment interface 160. Next, the results block 580 stores the data is stored in the data stores 230, 232 against the lot number and layer for the current context (thread) and updates the jeopardy stack.

Referring now to Figure 6, a simplified flow diagram of a method for integrating multiple controllers in accordance with another illustrative embodiment of the present invention is provided. In block 600, workpieces are processed in a plurality of tools. In block 610, a baseline control script is initiated for a selected tool of the plurality of tools (*e.g.*, by the control execution manager 150). After initiation of the baseline control script, the control executor 220 performs the remaining tasks. In block 620, a group of required control routines for the selected tool is identified. In block 630, control state information is retrieved related to previous control actions associated with the selected tool for the group of required control routines. In block 640, a first control routine from the group of required control routines is executed to generate a first control action. In block 650, the control state information associated with a second control routine from the group of required control routines is changed based on the first control action. In block 660, the second control routine is executed based on the modified control state information to generate a second control action.

The particular embodiments disclosed above are illustrative only, as the invention may be modified and practiced in different but equivalent manners apparent to those skilled in the art having the benefit of the teachings herein. Furthermore, no limitations are intended to the details of construction or design herein shown, other than as described in the claims below. It is therefore evident that the particular embodiments disclosed above may be altered or modified and all such variations are considered within the scope of the invention. Accordingly, the protection sought herein is as set forth in the claims below.

CLAIMS

1. A method for controlling a manufacturing system (100), comprising:
processing workpieces in a plurality of tools (120, 125);
initiating a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125);
providing context information for the baseline control script (152, 154);
determining a tool type based on the context information;
selecting a control routine for the selected tool (120, 125) based on the tool type; and
executing the control routine to generate a control action for the selected tool (120, 125).

2. The method of claim 1, wherein selecting the control routine further comprises linking to a library (240) of control routines.

3. The method of claim 1, wherein the context information comprises an entity identification code associated with the selected tool and determining the tool type further comprises determining the tool type based on the entity identification code.

4. The method of claim 1 or 3, wherein the context information comprises an operation identification code, and selecting the control routine further comprises selecting the control routine based on the tool type and the operation identification code.

5. The method of claim 1, 3, or 4, wherein the context information comprises a product identification code, and selecting the control routine further comprises selecting the control routine based on the tool type and the product identification code.

6. A manufacturing system (100), comprising:
a plurality of tools (120, 125) adapted to process workpieces;
a control execution manager (150) adapted to initiate a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125) and provide context information for the baseline control script (152, 154); and
a control executor (220) adapted to execute the baseline control script (152, 154), determine a tool type based on the context information, select a control routine for the selected tool (120, 125) based on the tool type, and execute the control routine to generate a control action for the selected tool (120, 125).

7. The system (100) of claim 6, wherein the control executor (220) is adapted to link to a library (240) of control routines to select the control routine.

8. The system (100) of claim 6, wherein the context information comprises an entity identification code associated with the selected tool, and the control executor (220) is adapted to determine the tool type based on the entity identification code.

9. The system (100) of claim 6 or 8, wherein the context information comprises an operation identification code, and the control executor (220) is adapted to select the control routine based on the tool type and the operation identification code.

10. The system (100) of claim 6, 8, or 9, wherein the context information comprises a product identification code, and the control executor (220) is adapted to select the control routine based on the tool type and the product identification code.

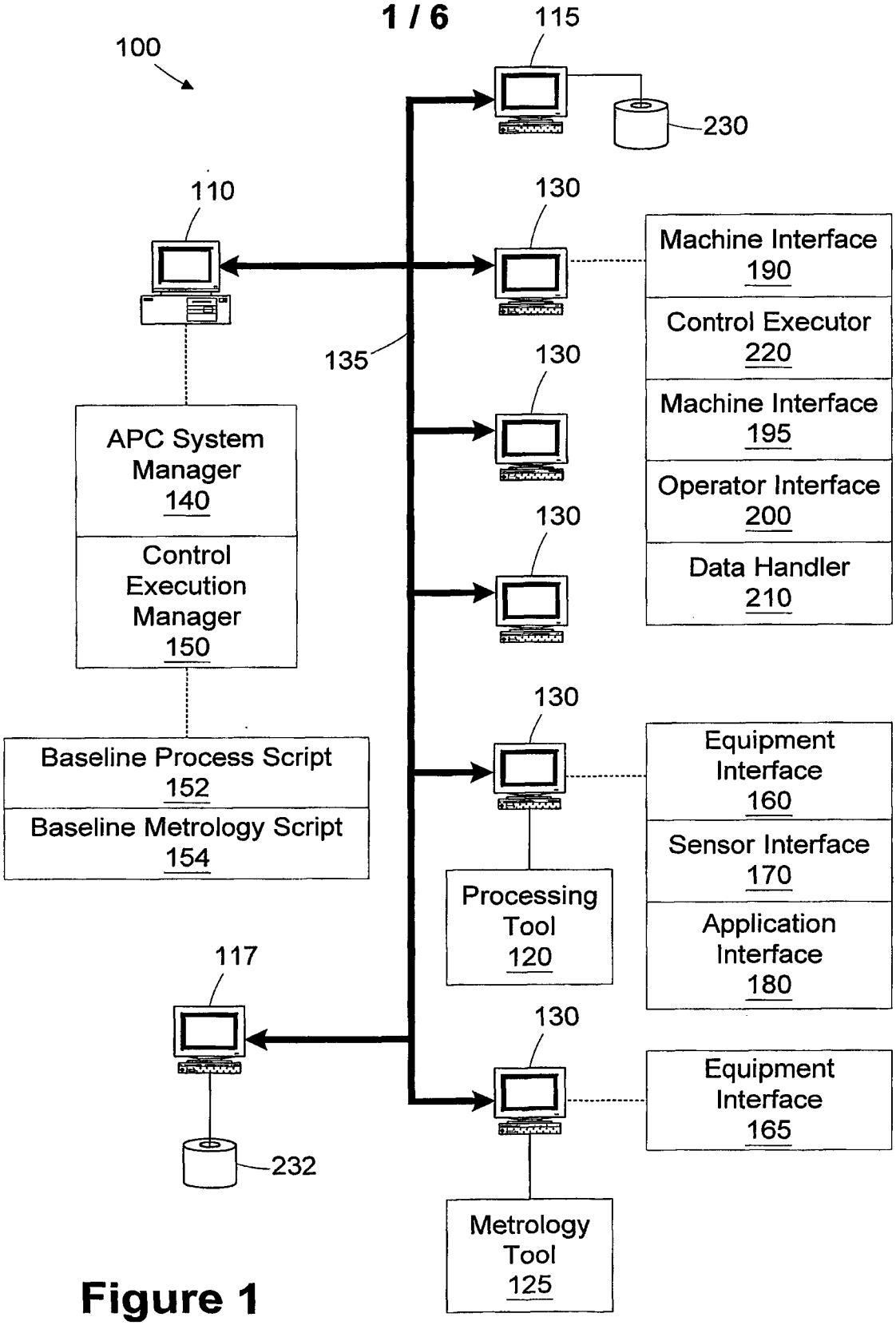

100
115
230
110
130
Machine Interface
190
Control Executor
220
Machine Interface
195
Operator Interface
200
Data Handler
210
135
130
130
APC System
Manager
140
Control
Execution
Manager
150
Baseline Process Script
152
Baseline Metrology Script
154
130
Equipment
Interface
160
Sensor Interface
170
Application
Interface
180
Processing
Tool
120
117
130
Equipment
Interface
165
232
Metrology
Tool
125

Figure 1

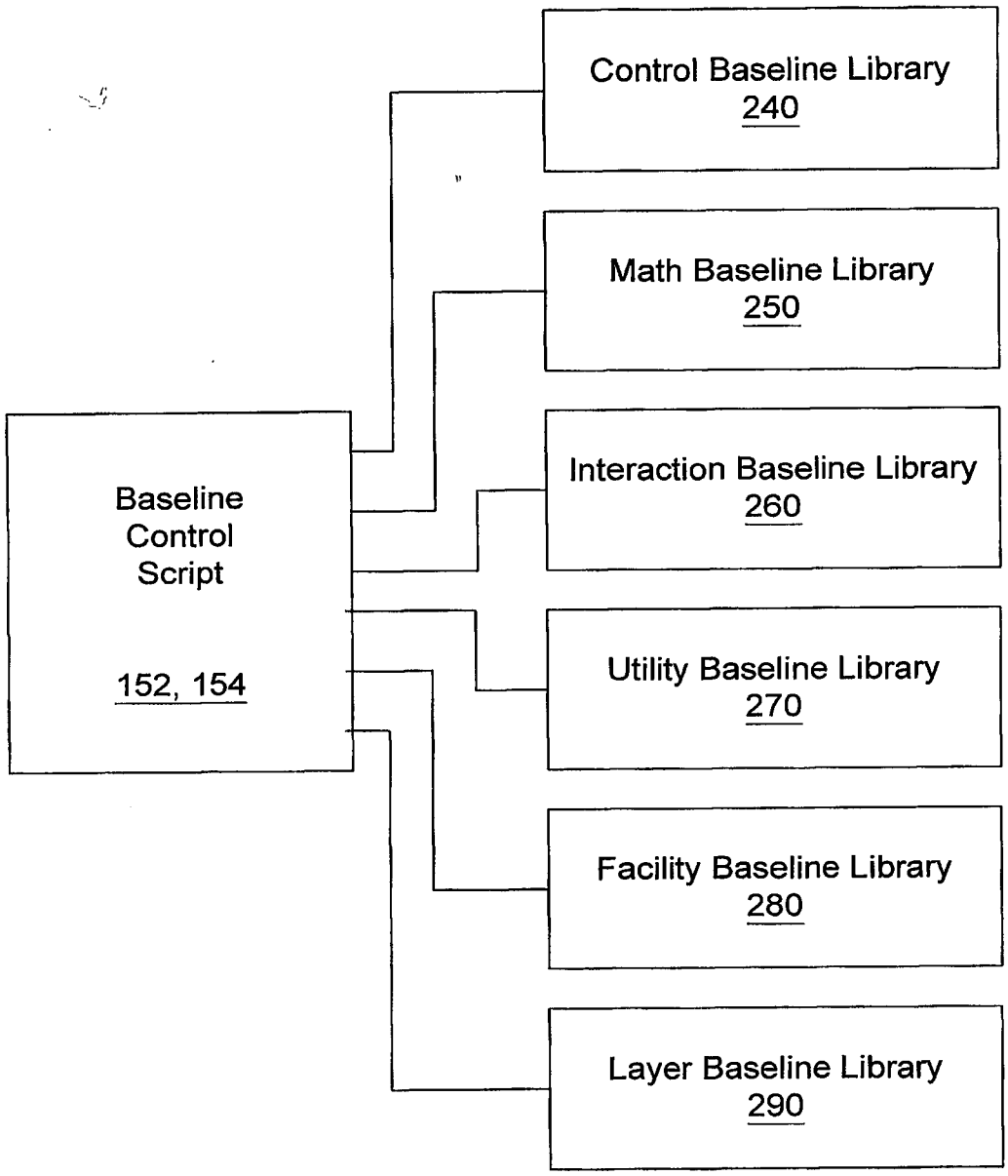

Control Baseline Library
240
Math Baseline Library
250
Interaction Baseline Library
260
Utility Baseline Library
270
Facility Baseline Library
280
Layer Baseline Library
290
Baseline
Control
Script
152, 154

Figure 2

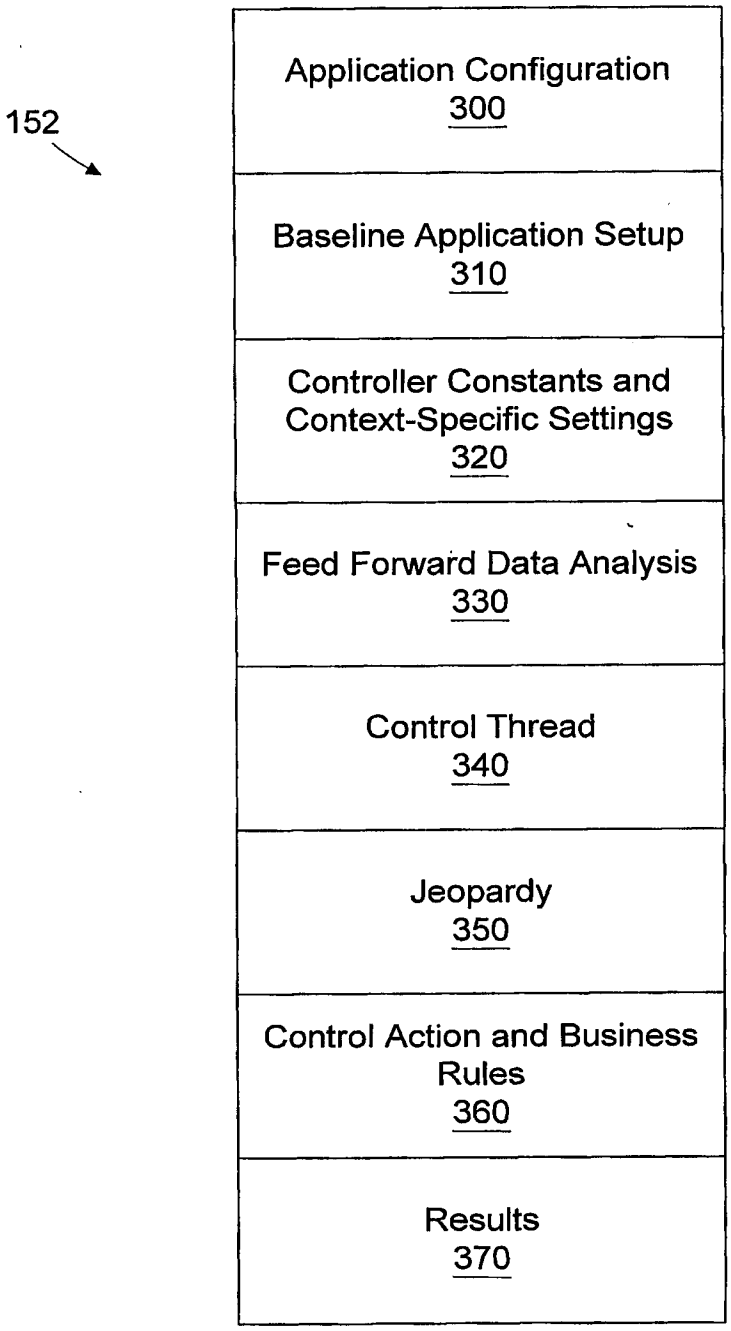

152
Application Configuration
300
Baseline Application Setup
310
Controller Constants and Context-Specific Settings
320
Feed Forward Data Analysis
330
Control Thread
340
Jeopardy
350
Control Action and Business Rules
360
Results
370

Figure 3

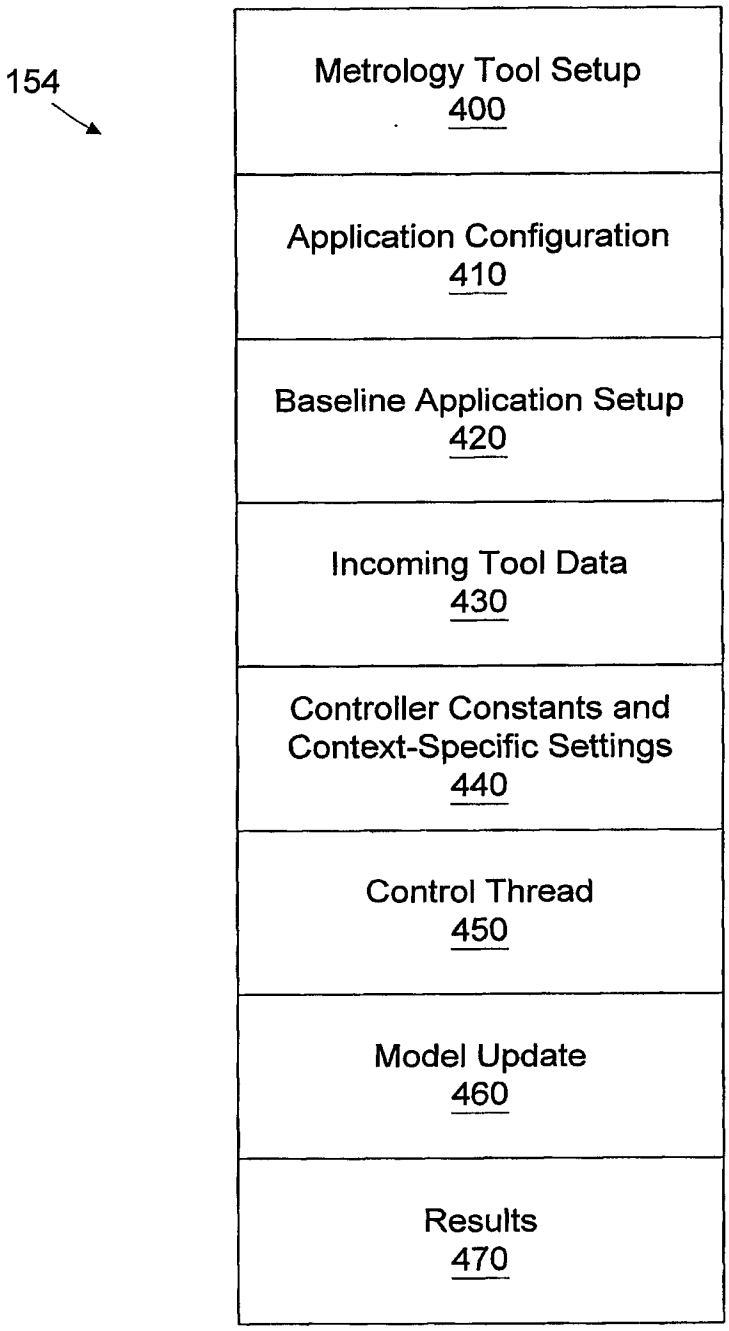

154
Metrology Tool Setup
400
Application Configuration
410
Baseline Application Setup
420
Incoming Tool Data
430
Controller Constants and
Context-Specific Settings
440
Control Thread
450
Model Update
460
Results
470

Figure 4

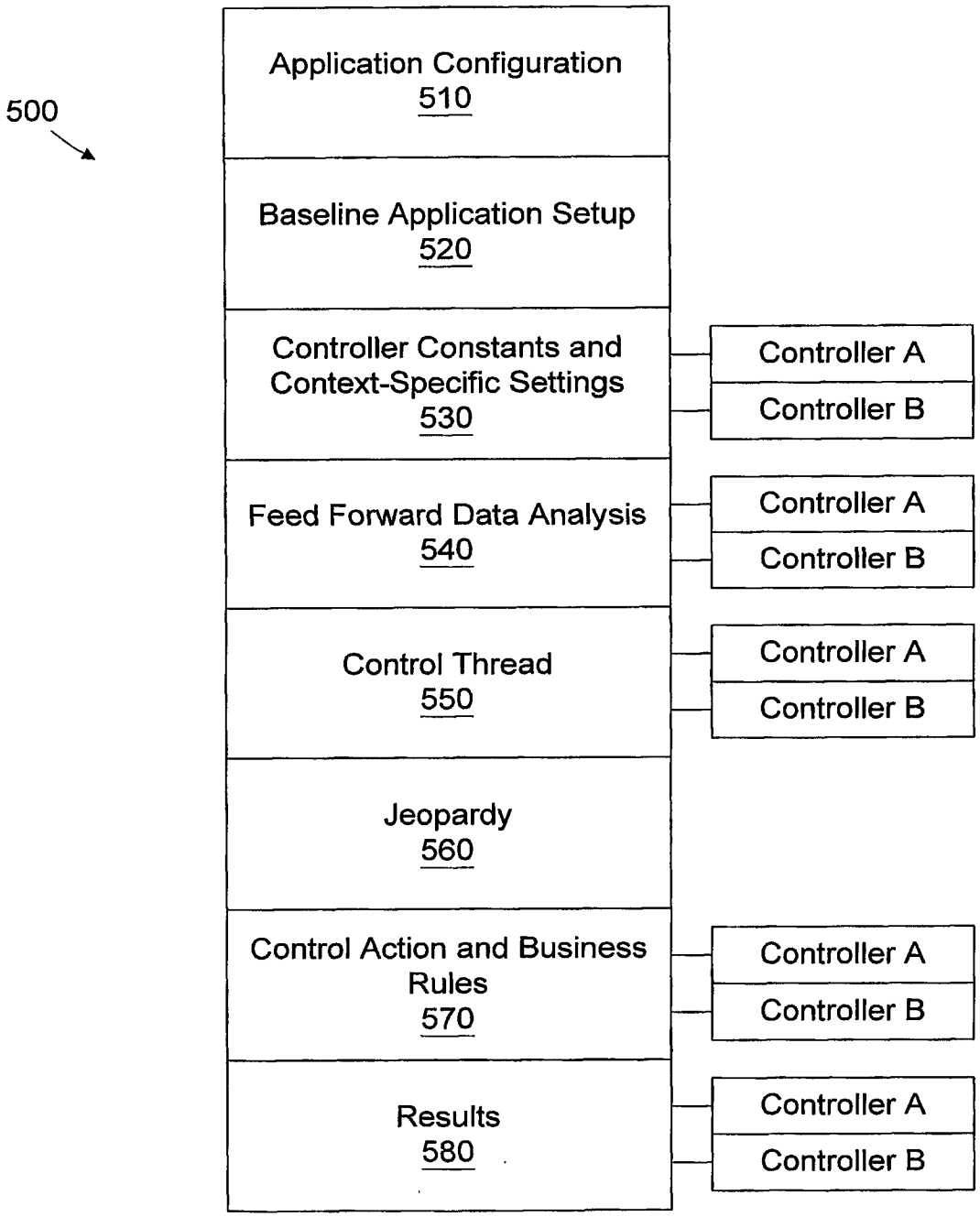

500
Application Configuration
510
Baseline Application Setup
520
Controller Constants and Context-Specific Settings
530
Feed Forward Data Analysis
540
Control Thread
550
Jeopardy
560
Control Action and Business Rules
570
Results
580
Controller A
Controller B
Controller A
Controller B
Controller A
Controller B
Controller A
Controller B
Controller A
Controller B

Figure 5

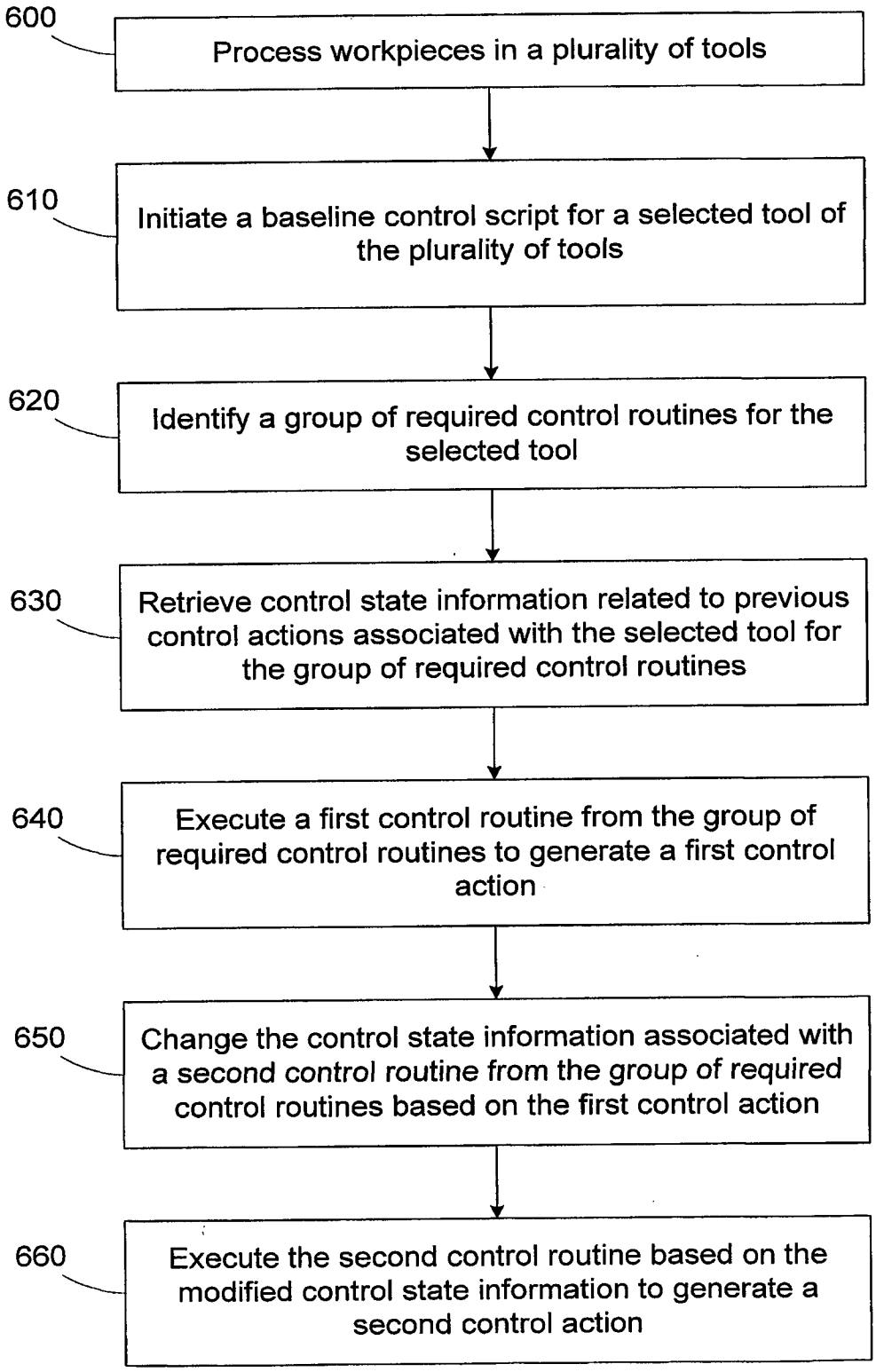

600
Process workpieces in a plurality of tools
610
Initiate a baseline control script for a selected tool of the plurality of tools
620
Identify a group of required control routines for the selected tool
630
Retrieve control state information related to previous control actions associated with the selected tool for the group of required control routines
640
Execute a first control routine from the group of required control routines to generate a first control action
650
Change the control state information associated with a second control routine from the group of required control routines based on the first control action
660
Execute the second control routine based on the modified control state information to generate a second control action

Figure 6

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069063 A3

(51) International Patent Classification[7]: G05B 19/418, 19/042

(21) International Application Number: PCT/US01/50178

(22) International Filing Date: 22 October 2001 (22.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/789,871 21 February 2001 (21.02.2001) US

(71) Applicant: **ADVANCED MICRO DEVICES, INC.** [US/US]; One AMD Place, Mail Stop 68, Sunnyvale, CA 94088-3453 (US).

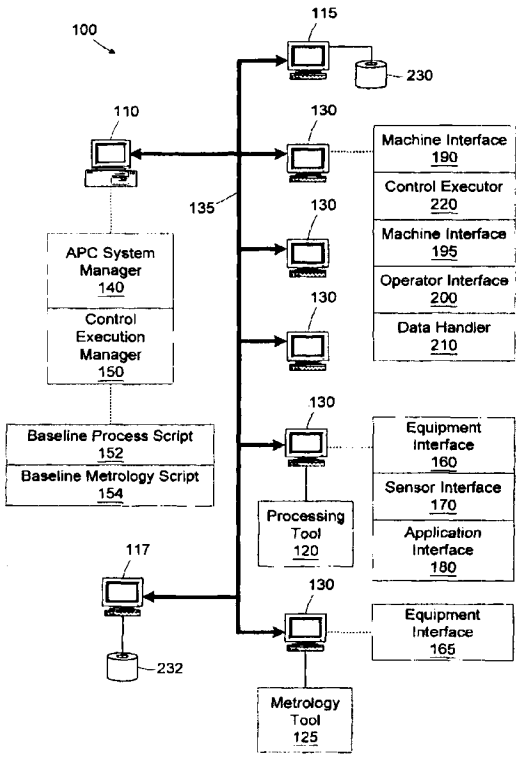
(72) Inventors: **BODE, Christopher, A.**; 4700 Staggerbrush Road #738, Austin, TX 78749 (US). **PASADYN, Alexander, J.**; 8717 Dandelion Trail, Austin, TX 78745 (US). **TOPRAC, Anthony, J.**; 4023 Walnut Clay, Austin, TX 78731 (US). **HEWETT, Joyce, S., Oey**; 6517 Sans Souci Cove, Austin, TX 78759 (US). **PETERSON, Anastasia, Oshelski**; 5811 Cannon Mountain Drive, Austin, TX 78749 (US). **SONDERMAN, Thomas, J.**; 16010 Braesgate Drive, Austin, TX 78717 (US). **MILLER, Michael, L.**; 2614 Little Elm Trail, Cedar Park, TX 78613 (US).

(74) Agent: **DRAKE, Paul, S.**; Advanced Micro Devices, Inc., 5204 East Ben White Boulevard, Mail Stop 562, Austin, TX 78741 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR CONTROLLING A TOOL USING A BASELINE CONTROL SCRIPT



(57) Abstract: A method for controlling a manufacturing system (100) includes processing workpieces in a plurality of tools (120, 125); initiating a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125); providing context information for the baseline control script (152, 154); determining a tool type based on the context information; selecting a control routine for the selected tool (120, 125) based on the tool type; and executing the control routine to generate a control action for the selected tool (120, 125). A manufacturing system (100) includes a plurality of tools (120, 125) adapted to process workpieces, a control execution manager (150), and a control executor (220). The control execution manager (150) is adapted to initiate a baseline control script (152, 154) for a selected tool (120, 125) of the plurality of tools (120, 125) and provide context information for the baseline control script (152, 154). The control executor (220) is adapted to execute the baseline control script (152, 154), determine a tool type based on the context information, select a control routine for the selected tool (120, 125) based on the tool type, and execute the control routine to generate a control action for the selected tool (120, 125).

WO 02/069063 A3

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
10 July 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Intern/ il Application No
PCT/US 01/50178

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G05B19/418 G05B19/042

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GB 2 329 041 A (MITSUBISHI ELECTRIC CORP) 10 March 1999 (1999-03-10) page 23 -page 44 figures 1-16	1,3-6, 8-10
Y		2,7
Y	US 4 698 766 A (ROBERTS PETER ET AL) 6 October 1987 (1987-10-06) column 2, line 1-11 column 4, line 48-52 column 8, line 52 -column 9, line 42 figures 1,2	2,7
A	WO 00 36479 A (SMITH RANDY ;HIER CRAIG ALAN (US); SPEEDFAM IPEC CORP (US)) 22 June 2000 (2000-06-22) page 1 -page 10; figures 1,2	1-10

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 April 2003	16/04/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	MESEGUER MAYORAL, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Interna I Application No
PCT/US 01/50178

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
GB 2329041 A	10-03-1999	JP 10086040 A	07-04-1998
		DE 19725101 A1	02-01-1998
		GB 2314174 A ,B	17-12-1997
		HK 1004961 A1	28-04-2000
		KR 252460 B1	15-04-2000
		US 5870306 A	09-02-1999
US 4698766 A	06-10-1987	DE 3581000 D1	07-02-1991
		EP 0162670 A2	27-11-1985
		GB 2158970 A ,B	20-11-1985
		JP 61019548 A	28-01-1986
WO 0036479 A	22-06-2000	TW 464796 B	21-11-2001
		WO 0036479 A1	22-06-2000